UNIVERSIDAD DE PANAMA
FACULTAD DE INFORMATICA ELECTRÓNICA Y COMUNICACIÓN

MAESTRIA EN CIENCIAS DE INGENIERIA DE SISTEMAS DE COMUNICACIONES CON ENFASIS EN REDES DE DATOS
AUTOMATIZACIÓN DE UN SISTEMA DE RIEGO AGRICOLA CON CÓDIGO ABIERTO

BLADIMIR JAIME PÉREZ QUEZADA

TESIS PRESENTADA EN CUMPLIMIENTO DE LOS REQUISITOS EXIGIDOS PARA OPTAR POR EL GRADO DE MAESTRIA EN CIENCIAS DE INGENIERIA DE SISTEMAS DE COMUNICACIONES CON ENFASIS EN REDES DE DATOS

PANAMA REPUBLICA DE PANAMA
2014

6 MAR 2015

PROFESOR ASESOR
MGTR JAVIER FERNANDEZ
DEPARTAMENTO DE ELECTRÓNICA Y COMUNICACIÓN

UNIVERSIDAD DE PANAMA
VICERRECTORIA DE INVESTIGACION Y POSTGRADO
DIRECCIÓN DE POSTGRADO

VIP DP-005 13
2 de enero de 2013

Profesor
Gustavo Díaz
Director de Investigación y Postgrado
Maestría en Ciencias de Ingeniería de Sistemas de Comunicaciones
con énfasis en Redes de Datos
Facultad de Informática Electrónica y Comunicación
Universidad de Panamá
E S D

Estimada Senor Director

Atendiendo su solicitud de inscripción del Proyecto de Tesis adjunto copia de la misma con su respectivo código para los trámites pertinentes

NOMBRE DEL ESTUDIANTE	TÍTULO DE LA TESIS	CÓDIGO
Bladimir J Pérez	Automatización de un Sistema de Riego Agrícola con código abierto	CE PI 327 17-02 14-05

Atentamente

Dr Heliberto Morales
Director de Postgrado

c c Dr Tomas Diez Director de Investigación VIP

/hed

UNIVERSIDAD DE PANAMA
VICERRECTORIA DE INVESTIGACION Y POSTGRADO

ACTA DE SUSTENTACIÓN DE TESIS

CÓDIGO CE PI 327 17-02 14-05
PROGRAMA DE MAESTRÍA EN MAESTRIA EN CIENCIAS EN INGENIERÍA DE COMUNICACIONES CON ÉNFASIS EN REDES DE DATOS

Título de la Tesis ***AUTOMATIZACIÓN DE UN SISTEMA DE RIEGO AGRÍCOLA CON CÓDIGO ABIERTO***

Nombre del Participante Bladimir Pérez

Cedula 2 717 101

Miembros del Jurado	Calificación otorgada
Javier Hernandez	100
Alva o Maturell	100
Gustavo Díaz	100

Firma de los Miembros del Jurado	Trabajo Escrito	Defensa	Promedio
Alva Maturell	100	100	100
Javier Hernández	100	100	100
	100	100	100
		NOTA FINAL	100

Recomendaciones del Jurado _____

Firma del Director de I P ó Coordinador del Programa de _____

Firma del Representante de la Vicerrectoría de Investigación y Postgrado (VIP) _____

Firma del Estudiante Bladimir Pérez

Fecha 28 3 2014

AGRADECIMIENTOS

Primeramente darle gracias a Dios por darme lo necesario para culminar esta nueva meta en mi vida y tener siempre a mi familia para apoyarme

Gracias a SENACYT por la beca otorgada a mi persona para cubrir todos los gastos relacionados al cumplimiento de esta Maestria

Agradecerle a la Universidad de Panama y sus docentes por transmitirme sus conocimientos y asi poder culminar esta meta principalmente al MGTR Javier Fernandez que como accesor supo guiarme para obtener este nuevo titulo universitario

Al centro de investigaciones de las tecnologias de información y comunicación de la universidad de panamá (CITIC UP) por facilitarme el uso de los módems XBee que se utilizaron en esta investigación también a su directo Dr Ivan Armuelles por todo el apoyo brindado en el tema de la pasantia

A todo el grupo del laboratorio de sistemas inteligentes de la Universidad Carlos III de Madrid (LSI UC3M) por todo el apoyo obtenido durante los tres meses de estancia que estuve desarrollando mi investigacion en sus instalaciones

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INDICE DE CUADROS

RESUMEN

En el presente trabajo se desarrollo un prototipo para ser implementado en un sistema de riego agricola para lograr su automatización monitoreo y control remotamente Para ello se utilizó herramientas de bajo costo flexibilidad y respaldo como lo son Arduino Android y XBee Arduino y XBee se encargaran de automatizar el sistema Android logrará el monitoreo y control remoto desde cualquier parte del mundo donde exista servicio de redes celulares e Internet De esta forma lograremos brindarle al agricultor la comodidad y seguridad que no obtiene con un sistema controlado de forma manual

SUMMARY

In this job was developed a prototype to be implemented in an agricultural irrigation system to achieve their automation monitoring and control remotely For this were used inexpensive tools as Android Arduino and XBee Arduino and XBee were responsible of automating the system Android achieved the remote monitoring and control from anywhere in the world where service cellular networks and the Internet exists In this way we can give the farmer the comfort and security that he do not get with a manually controlled system

1 ASPECTOS GENERALES

1 1 Introducción

La automatización nacio con el fin de usar la capacidad de las máquinas para llevar a cabo determinadas tareas anteriormente efectuadas por seres humanos y para controlar la secuencia de las operaciones sin intervención humana En comunicaciones aviación astronáutica y el sector industrial se han estado utilizando dispositivos como los equipos automaticos de conmutacion telefonica los pilotos automáticos de guia y los sistemas automatizados de control para efectuar diversas tareas con mayor rapidez y precisión mejor de lo que podria hacerlo un ser humano

La automatización para la industria es usar tecnologia que integre un proceso de control a través de dispositivos capaces de tomar decisiones e interactuar con otros basándose en un programa establecido por el integrador para el manejo de algunas variables mediante su monitoreo y comparación con un valor esperado del proceso esto se realiza de manera automática generando en el sistema mayor productividad confiabilidad estabilidad y calidad en sus resultados

Un proyecto de automatización se inicia cuando se identifica una oportunidad de mejorar procesos productivos susceptibles de ser automatizados Tal oportunidad puede ser un incremento en la producción el perfeccionamiento en los atributos y cualidades de alguna línea de productos para enfrentar la competencia de otros proveedores o lo más

comun mantener la fabricación y calidad dentro de las normas actuales pero disminuyendo los costos totales asociados a la producción

La automatización de los procesos productivos se establece como una herramienta fundamental que permite un desarrollo propio dinámico y competitivo facilitando la relación entre las diferentes áreas de organizaciones o empresas

Automatizar de un sistema de riego depende de lo cuidadoso que sean los implementadores en todas las fases de diseno instalación operación y mantenimiento con la finalidad de que el agua llegue a todos los puntos del área de producción agricola

A nivel mundial existen diferentes tipos de riego manual los cuales son empleados en diferentes tecnicas de mejora agricola a favor de la producción de cultivos tradicionales y no tradicionales Los diferentes tipos de riego manual son

- Riego por goteo (también llamado riego localizado)
- Riego por aspersión
- Riego superficial el cual se divide en las siguientes modalidades
    - Por inundación
    - Por surcos
    - Por melgas y otros

Estos sistemas de riego mencionados anteriormente son operados de forma manual por este motivo el agricultor no tiene la certeza de cuánta agua necesita el cultivo el cultivo necesita de riego existe exceso de agua por motivos ambientales y no se necesita regar el sistema de bombeo tiene problemas Las respuestas a estas preguntas las puede proporcionar un sistema automatizado en este caso un riego agrícola Además desarrollando una aplicación móvil capaz de ser ejecutada en un dispositivo móvil

(Laptop SmartPhone Tablet) podemos recibir alertas en todo momento como por ejemplo inicio o finalizo el riego la bomba del sistema de riego no encendió exceso de agua en el cultivo Con esta investigación logramos desarrollar un prototipo que supervisa y mantiene un control en todo momento del sistema de riego sin necesidad de visitar el cultivo diariamente como se acostumbra esto nos da seguridad flexibilidad facilita el control integral de los sistemas para los usuarios prepara el camino para nuevos servicios y también nos brinda un ahorro en los recursos utilizados

1 2 Definición del Problema

En la actualidad los sistemas de riego en su gran mayoria son operados de forma manual requiriendo mucha atención del agricultor Básicamente el agricultor por su experiencia decide el periodo de tiempo y el momento de regar el cultivo Debido a esto se pueden presentar problemas comunes generados por el ser humano como por ejemplo olvidar encender y apagar el sistema en el tiempo considerado Lo anterior causa problemas directos con el cultivo como por ejemplo exceso de agua al cultivo lo que acarrearia mayor consumo del recurso hidrico del pais que es tan importante a nivel nacional e internacional o bien secándolo por falta de la misma además de un mayor consumo energético del sistema Todo lo anterior disminuye la productividad causándole pérdidas económicas al agricultor Teniendo un sistema automatizado se busca solucionar estos errores aumentar la producción ahorrar en el consumo energetico y minimizar la merma en la producción ya que el agricultor no tendrá que estar pendiente permanentemente del cultivo porque el sistema al ser autónomo tomaria las decisiones requeridas Además si al desarrollo se le añade la opción de monitoreo y control remoto a

través de una aplicación móvil el agricultor se sentirá cómodo y seguro de que su cultivo está siendo regado adecuadamente

1.3 Hipótesis

Un prototipo de un sistema agrícola automatizado será económico flexible amigable al agricultor monitoreado y controlado remotamente además de escalable para futuras investigaciones

1 4 Objetivos

- Desarrollar un prototipo de un sistema de riego agrícola automatizado capaz de ser monitoreado y controlado remotamente desde una aplicación movil
- Elegir el micro-controlador sensores tecnologia inalámbrica y demás herramientas que nos ayuden a desarrollar la investigación
- Utilizar herramientas de bajo costo y alta flexibilidad
- Programar en lenguajes de acceso libre alta disponibilidad y escalabilidad
- Desarrollar una aplicación de ejecución en dispositivos móviles como Tablets SmartPhone capaz de comunicarse remotamente con el micro controlador

1 5 Metodología

Actualmente en el mercado existe una variedad de micro-controladores con grades capacidades para administrar procesos de automatización los cuales son bastante costosos y en la mayoria de los casos no manipulables en su parte lógica, permitiendo solamente manipularlos por lenguajes de programación propietarios por los fabricantes de

los mismos limitando de esta forma el desarrollo de programaciones lógicas con lenguajes de programación de libre acceso impidiendo el acceso para poder implementar nuevas aplicaciones como comunicaciones remotas con un dispositivo movil Por este motivo es tan importante investigar cual es el micro-controlador en el mercado capaz de ser programado tanto por lenguajes propietarios como por lenguajes de acceso libre que la relación costo beneficio sea la mejor para el proceso que en este caso es un sistema de riego agricola Para lograr este objetivo se estudiará a fondo los diferentes fabricantes el soporte técnico de los mismos los más usados en aspectos de investigacion y desarrollo lo anterior garantiza un acceso libre al nucleo de la lógica del micro controlador afirmando las facilidades para el desarrollo libre e implementación

El micro controlador deberá tener multiples entradas y salidas para controlar bombas de riego válvulas de agua, sensores de nivel de agua sensores de humedad puertos de comunicaciones El mercado actual cuenta con micro-controladores que cumplen con estas caracteristicas y se procederá a elegir uno de ellos Elegido el micro controlador se procederá a programar el mismo con rutinas de control para luego verificar su comportamiento y hacer los ajustes necesarios Al mismo se le conectaran terminales ficticias en sus puntos de entrada y salida con el objetivo de simular bombas de riego válvulas de agua pero si se manejaran de forma real los sensores de humedad ya que esto será el punto clave para el buen funcionamiento del sistema Se tomaran datos y se corregirán probables errores en la simulación

Logrado el funcionamiento óptimo y autonomo del micro controlador procederemos a desarrollar una aplicación móvil capaz de enviar y recibir datos del micro-controlador a través de la red de telefonia celular La aplicación móvil recibirá los datos entregados por

el micro controlador y los administrará de forma tal que el usuario los pueda entender de una forma clara cómoda y amigable

Desarrollada la aplicación móvil se procederá a realizar pruebas de comunicación entre el micro-controlador y la aplicación móvil que se ejecutara en diferentes dispositivos móviles como Tablets y SmartPhone Con cada uno de los dispositivos móviles se realizaran pruebas observando el rendimiento de la aplicación en cada uno de ellos documentando los resultados obtenidos Se realizaran los cambios pertinentes si se requieren para lograr obtener el mejor desempeño de la aplicación móvil en un SmartPhone debido a la popularidad comodidad y accesibilidad de este dispositivo en la sociedad actual Todas las pruebas antes descritas se realizaran en un entorno controlado de laboratorio

1 6 Sector Beneficiado

Con la investigación se beneficiará el sector agricola obteniendo mejores cosechas además se hará un buen uso de los recursos hídricos utilizando solo el necesario del mismo para los cultivos esto repercute directamente en un menor consumo de energia utilizado por el sistema de bombeo

2 FUNDAMENTACIÓN TEÓRICA

2 1 Tipos de Riego

En la actualidad se utilizan diferentes tipos de riego siendo los más comunes en Panamá los mencionados a continuación riego por superficie aspersión localizado (Edward y Martin (2010))

2 1 1 Riego por Superficie

Son riegos muy conocidos que en principio no crean problemas al agricultor experto pero que pueden producir pérdidas de abonos por lavados y arrastres al no poder controlarse perfectamente las dosis de agua

El agua se aplica directamente sobre la superficie del suelo por gravedad El propio suelo actua como sistema de distribución dentro de la parcela desde la zona próxima al lugar de suministro denominado cabecera de la parcela, hasta llegar a todos los puntos de ella Finalmente el agua alcanza la cola de la parcela

El agua puede llegar hasta la parcela por medio de cualquier sistema de distribución bien por tuberías o por una red de canales donde el agua circula por gravedad Una vez que el agua está en cabecera no es preciso dotarla de presión ya que se vierte sobre el suelo y fluye libremente lo que supone evitar tener en la parcela un complejo sistema de tuberias y piezas especiales para distribuir el agua a presión así como un ahorro de

energia ya que no se precisan sistemas de bombeo Para distribuir el agua adecuadamente es muy frecuente disponer de surcos que favorezcan la circulación o escurrimiento del agua sobre el suelo a lo que también contribuye la pendiente que suelen tener las parcelas de riego en la dirección de escurrimiento del agua aun cuando existen parcelas a nivel en las que la pendiente es cero

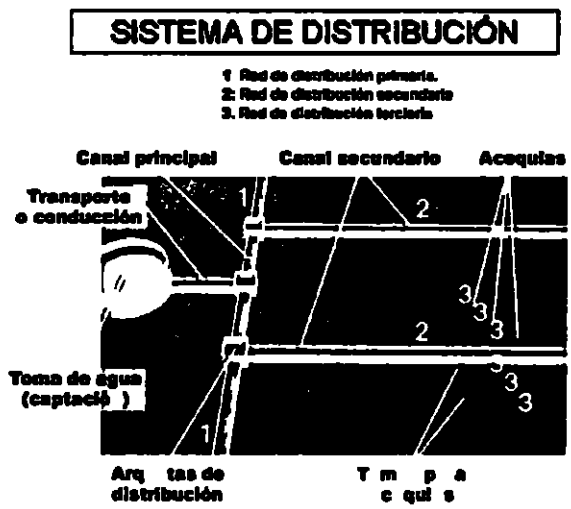


Fig 1 Distribución de un sistema de riego por superficie

El riego por superficie es un método particularmente recomendable en terrenos llanos o con pendientes muy suaves en las que no sea preciso realizar una explanación del suelo que es costosa y puede afectar negativamente al suelo Es el metodo de riego menos costoso en instalación y mantenimiento y una vez que el agua llega a la parcela no existe coste en la aplicación del agua Es con diferencia el sistema de riego que utiliza el agua de forma menos eficiente

Dada la gran variedad de sistemas diferentes dentro de la aplicación del agua por gravedad el riego por superficie puede aplicarse casi a la totalidad de los cultivos (Edward y Martin (2010))

2 1 2 Riego por Aspersión

Con este método el agua se aplica al suelo en forma de lluvia utilizando unos dispositivos de emisión de agua denominados aspersores que generan un chorro de agua pulverizada en gotas El agua sale por los aspersores dotada de presión y llega hasta ellos a través de una red de tuberias cuya complejidad y longitud depende de la dimensión y la configuración de la parcela a regar Por lo tanto una de las caracteristicas fundamentales de este sistema es que es preciso dotar al agua de presión a la entrada en la parcela de riego por medio de un sistema de bombeo La disposición de los aspersores se realiza de forma que se moje toda la superficie del suelo de la forma más homogénea posible

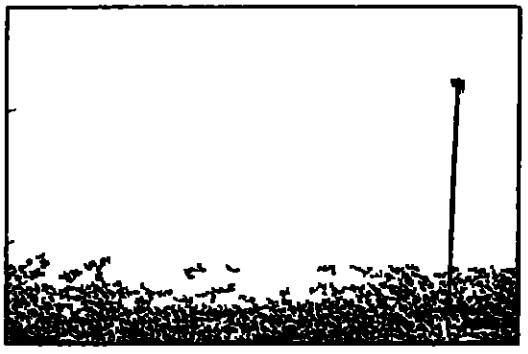


Fig 2 Riego por aspersión

Un sistema de riego tradicional por aspersión está compuesto de tuberias principales (normalmente enterradas) y tomas de agua o hidrantes para la conexión de secundarias ramales de aspersión y los aspersores Todos o algunos de estos elementos pueden estar fijos en el campo permanentes o solo durante la campaña de riego Además también pueden ser completamente móviles y ser transportados desde un lugar a otro de la parcela

Los sistemas de riego por aspersión se adaptan bastante bien a topografías ligeramente accidentadas El consumo de agua es moderado y la eficiencia de uso bastante aceptable Sin embargo la aplicación del agua en forma de lluvia está bastante condicionada a las condiciones climáticas que se produzcan en particular al viento y a la aridez del clima, ya que si las gotas generadas son muy pequeñas en particular el viento y a la aridez del clima las gotas podrían desaparecer antes de tocar el suelo por la evaporacion

Son especialmente utiles para aplicar riegos relativamente ligeros con los que se pretende aportar algo de humedad al suelo en el periodo de nacencia o para aplicar riegos de socorro También es muy indicado para efectuar el lavado de sales cuando sea necesario y se prestan a la aplicación de determinados productos fitosanitarios o abonos disueltos en el agua de riego aunque no se puede considerar que sea una aplicación habitual

2 1.3 Riego Localizado

El riego localizado consiste en aplicar agua a una zona determinada del suelo no en su totalidad Al igual que en el riego por aspersión el agua circula a presión por un sistema de tuberías (principales secundarias terciarias y ramales) desplegado sobre la superficie del suelo o enterrado en este saliendo finalmente por los emisores de riego localizado con poca o nula presión a través de unos orificios generalmente de muy pequeño tamaño

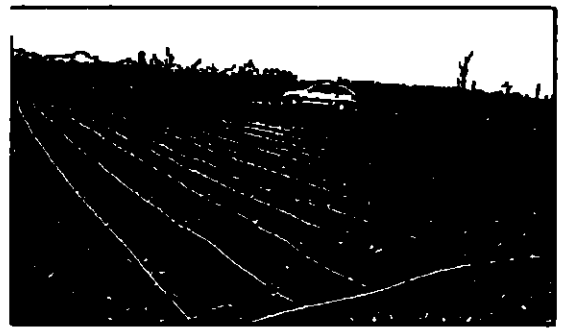


Fig 3 Riego localizado

En estos sistemas es necesario contar con un sistema de bombeo que dote de presión al agua, así como determinados elementos de filtrado y tratamiento del agua antes de que circule por la red de tuberías Con ellos se pretende evitar la obturación de los emisores uno de los problemas más frecuentes Estos elementos se instalan a la salida del grupo de bombeo en el denominado cabezal de riego

Es el sistema ideal para poner en práctica las técnicas de fertilizantes disueltos en el agua de riego El desarrollo de las técnicas y equipos han permitido una automatización de las instalaciones en distintos grados llegándose en ocasiones a un funcionamiento casi autónomo de todo el sistema De esta forma se consiguen automatizar operaciones como limpieza de equipos apertura o cierre de válvulas fertilización etc que producen un importante ahorro de mano de obra

Es el método de riego más tecnificado y con el que más fácil se aplica el agua de manera eficiente De igual forma, el manejo del riego es muy diferente del resto de los sistemas ya que el suelo pierde importancia como almacén de agua Se riega con bastante frecuencia para mantener un nivel óptimo de humedad en el suelo

Requiere un buen diseño una alta inversión en equipos y mantenimiento concienzudo es decir tiene un alto coste que pude ser asumido en cultivos de alto valor comercial

2 2 Métodos y Sensores Para Medir la Humedad del Suelo

El manejo apropiado del riego requiere la evaluación de parte del agricultor de sus necesidades de riego en base a medidas de varios parámetros físicos del suelo Algunos productores utilizan equipos sofisticados mientras que otros se basan en métodos empíricos o en el sentido comun Cualquiera que sea el método usado cada uno tiene sus propios méritos y limitaciones

El agricultor generalmente se hace dos preguntas al desarrollar una estrategia para el manejo del riego ¿Cuándo regar? y ¿Cuánta agua aplicar?

Los siguientes metodos se pueden utilizar para determinar el contenido de humedad del suelo (Glaria y Kouro (2001))

2 2 1 El Método del Tacto

La determinación de la humedad del suelo por medio del tacto ha sido utilizada por muchos años por investigadores y agricultores por igual Al apretar la tierra entre el pulgar y el dedo indice o al exprimir la tierra en la palma de la mano se puede obtener una estimación bastante aproximada de la humedad en el suelo Toma un poco de tiempo y algo de experiencia lograr esto pero es un método comprobado (Glaria y Kouro (2001))

2 2 2 La Sonda de Neutrones

Se han utilizado extensamente en trabajos de investigacion para determinar la humedad del suelo Contienen una fuente radioactiva que envia una cierta cantidad de

neutrones rápidos que son aproximadamente del tamaño de un átomo de hidrógeno un componente esencial del agua Cuando los neutrones rápidos chocan contra los átomos de hidrógeno se vuelven más lentos Un detector dentro de la sonda mide la proporción de los neutrones rápidos que salen y de los lentos que regresan Esta relación se usa entonces para estimar el contenido de humedad en el suelo (Glaria y Kouro (2001))

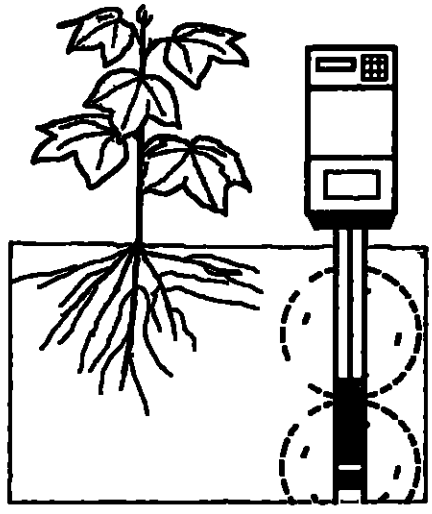


Fig 4 Sonda de Neutrones

2 2.3 La Resistencia Eléctrica

Otro método que ha sido utilizado por muchos años para determinar el contenido de humedad en el suelo es la medición de la resistencia eléctrica El principio físico de estos dispositivos es que el contenido de humedad se puede determinar por la resistencia al paso de corriente eléctrica entre dos electrodos en contacto con el suelo Entre más agua haya en la tierra más baja es la resistencia.

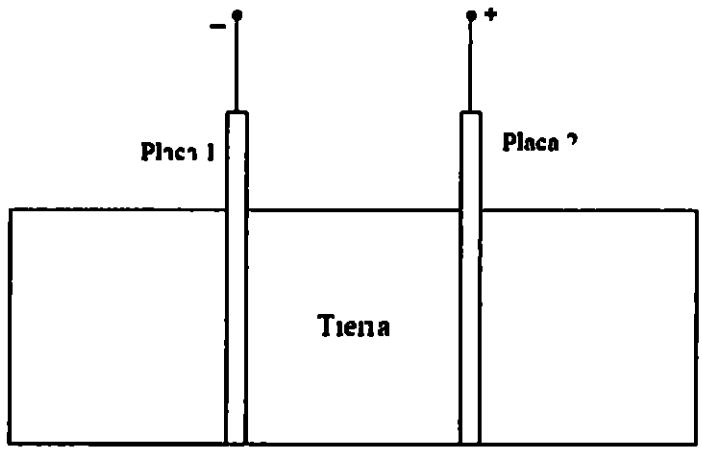


Fig 5 Esquema del sensor resistivo (Perera (2010))

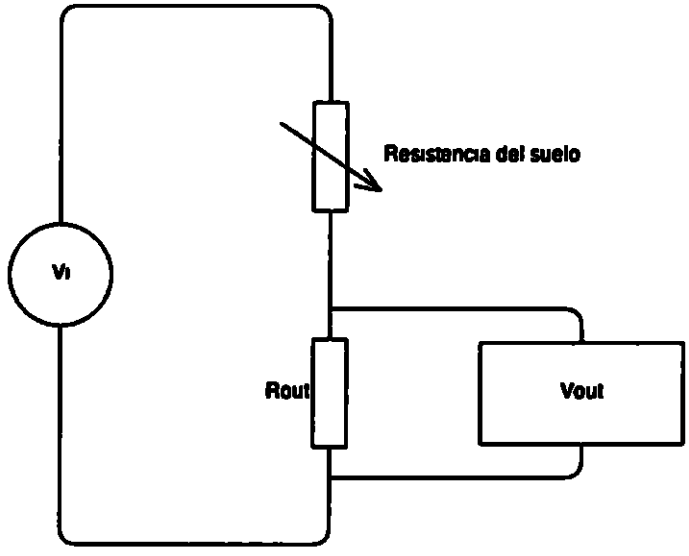


Fig 6 Circuito electrónico de un sensor resistivo

Gracias a las leyes de Kirchhoff podemos deducir la ecuación (5) que describe el comportamiento del circuito mostrado en la (Fig 6)

(1) $$V_{out} = I * R_{out}$$

(2) $$V_i = I * R_{suelo} + I * R_{out}$$

(3) $$V_i = I(R_{suelo} + R_{out})$$

(4) $$I = \frac{V_i}{R_{suelo\ R_o\ t}}$$

(5) $$V_{out} = \frac{V_t \ R_{out}}{R_{su\ t} + R_{out}}$$

Donde

V es el voltaje de entrada

$V_{o\ t}$ voltaje de salida, es el que podemos tomar para determinar la humedad

I corriente total que circula por el circuito

$R_{s\ elo}$ resistencia del suelo que varia segun el contenido de agua

R_{out} resistencia de las terminales de salida

Observando la ecuación (5) podemos decir que a medida que la resistencia del suelo disminuye por el contenido de agua el voltaje de salida aumenta El voltaje de salida es inversamente proporcional a la resistencia del suelo

2 2 4 Sensores Capacitivos

Son quizás los más difundidos en la industria pues son de fácil producción bajos costos y alta fidelidad El principio en el cual se basa este tipo de sensores es en el cambio que sufre la capacidad de un condensador al variar la constante dieléctrica del mismo En la formula (6) se muestra la relación de las variables

(6) $$C = \varepsilon \frac{A}{d}$$

Donde

C es el valor de la capacidad

ε es la constante dieléctrica

A el área de las placas del condensador

d la distancia entre las placas del condensador

Si se utiliza como dieléctrico una mezcla gaseosa que contenga vapor de agua, el valor C del condensador va a variar dependiendo de la cantidad de moléculas de agua que estén presentes entre las placas En consecuencia basta medir o convertir el cambio de capacitancia a otro tipo de variable eléctrica más fácil de manejar Lo anterior se puede lograr con un puente de Wheatstone de condensadores un circuito resonante o también utilizar el condensador como componente de un oscilador estable que varia su frecuencia de acuerdo al cambio de C (Perera (2010))

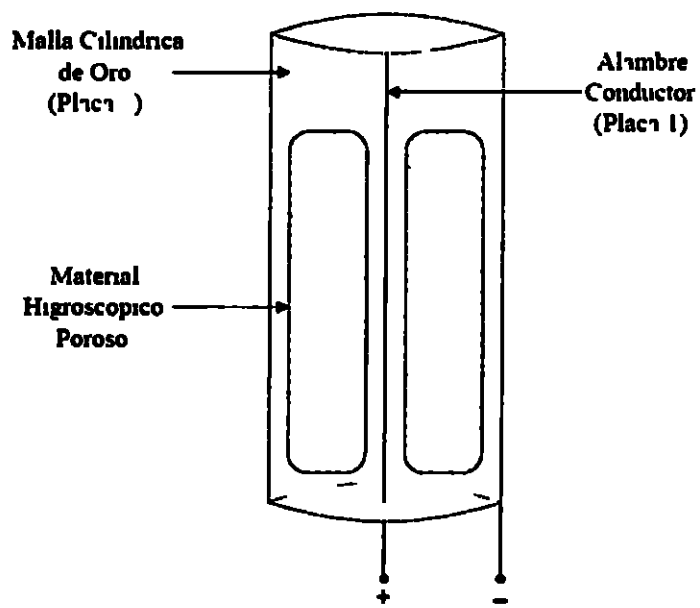


Fig 7 Sensor Capacitivo (Perera (2010))

2 2 5 La Tensión del Suelo

A medida que el suelo se seca las particulas del suelo retienen el agua con mayor fuerza Los tensiómetros miden la intensidad de la fuerza con la que el suelo retiene el agua La mayoría tiene una punta cerámica o porosa conectada a una columna de agua A medida que el suelo se seca comienza a jalar agua de la columna de agua a través del

bulbo de cerámica, provocando succion en la columna de agua Esta fuerza se mide entonces con un indicador de succión Algunos modelos más nuevos han reemplazado el indicador de succión con un sensor electrónico los cuales usualmente son más sensibles que los indicadores de aguja

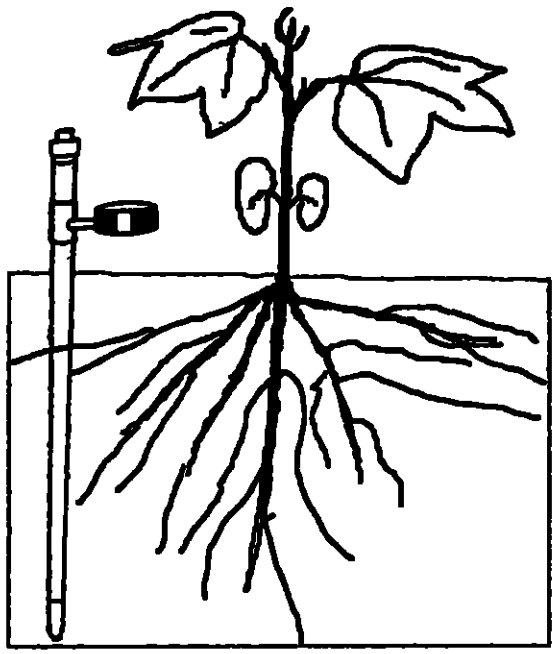


Fig 8 Diagrama de un tensiómetro

2 2 6 Reflectómetro de Dominio de Tiempo

Los instrumentos TDR (Reflectómetro de dominio de tiempo) funcionan bajo el principio de que la presencia de agua en el suelo afecta la velocidad de propagación de una onda electromagnética (la hace más lenta) El mismo envia una onda electromagnética a través de una guia (generalmente un par de puntas paralelas de metal) colocada en el suelo a la profundidad deseada entonces mide el tiempo que le toma a la onda viajar por la guia hacia el suelo y regresar Este aparato registra el tiempo y lo

convierte a una lectura de humedad del suelo Entre más mojado esté el suelo más tiempo le toma a la onda magnética viajar por el suelo y regresar por la guia

2 2 7 Sonda C y Reflectómetro de Dominio de Frecuencia

Utilizan un oscilador de corriente alterna para formar un circuito eléctrico en conjunto con el suelo Después de insertar las sondas que pueden ser puntas paralelas o anillos de metal en el suelo el oscilador produce valores de frecuencia de acuerdo al contenido de humedad en el suelo

2 2 8 Las Plantas Como Indicadores

Las plantas también son utiles para indicarnos cuando regar Las mismas le permiten al agricultor buscar directamente en ellas señales que le indiquen cuándo regar y no basarse en parámetros indirectos como el suelo o evaporación Al observar sus características se puede tener una buena idea del contenido de humedad en el suelo (Glaría y Kouro (2001))

2 2 9 Termómetros de Luz Infrarroja

Mide la temperatura térmica de la planta particularmente de las hojas del cultivo De igual forma que las personas sudan para mantenerse frescas las plantas transpiran a través de unas aberturas llamadas estomas Una vez comienzan a sufrir de estrés hidrico comienzan a cerrar sus estomas y dejan de transpirar provocando que la planta se caliente y la temperatura de las hojas aumente Con las lecturas de rayos infrarrojos se puede detectar este aumento de temperatura

2 3 ZigBee

ZigBee es un estándar abierto para radiocomunicaciones de baja potencia basado en la especificación IEEE 802 15 4 de redes inalámbricas de área personal o WPAN (wireless personal area network) Incluye un robusto y confiable protocolo de red de bajo consumo y coste con servicios de seguridad y garantiza la interoperabilidad entre dispositivos Trabaja a una velocidad de datos de 250 Kbps sobre la banda libre de 2 4 Ghz aunque también soporta las bandas de 868 y 900 MHz (Digi International (2008))

El estándar ZigBee es publicado y administrado por un grupo de empresas y fabricantes denominado ZigBee Alliance Por lo tanto el termino ZigBee es en realidad una marca registrada de este grupo y no un estándar técnico Sin embargo para fines no comerciales la especificación ZigBee se encuentra disponible de forma abierta y libre al publico

Las principales caracteristicas de ZigBee se mencionan a continuación

Bajo coste Estándar abierto que hace innecesario el pago de patentes

Bajo consumo energético Permite prolongar ampliamente la vida de las baterias

Sencillez Cuenta con una pila de protocolos de reducido tamaño

Alta confiabilidad Redes malladas con redundancia de enlaces y canales además de la posibilidad de enrutamiento alternativo automático

Despliegue de red sencillo Fácil de montar y administrar gracias a las redes ad hoc y al enrutamiento automático

Alta seguridad Proporciona integridad de datos y autenticacion haciendo uso del algoritmo de cifrado AES 128

Compatibilidad Al tratarse de un estándar abierto se garantiza la interoperabilidad entre dispositivos de diferentes fabricantes

Baja latencia Los retardos producidos en las distintas tareas de red son muy reducidos del orden de milisegundos frente a varios segundos en otras tecnologías

Gran capacidad Una misma red puede soportar más de 65 000 nodos independientes

A continuación en el (Cuadro 1) se muestra una comparativa entre las diferentes tecnologías inalámbricas

Cuadro 1 Comparativa de tecnologías inalámbricas (Montesinos (2013))

Standard	Bluetooth	UWB	ZigBee	Wi-Fi
IEEE spec.	802 15 1	802 15.3a	802 15 4	802.11a/b/g
Frequency band	2 4 GHz	3 1 10 6 GHz	868/915 MHz; 2 4 GH	2.4 GHz; 5 GHz
Max signal rate	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Nominal range	10 m	10 m	10 100 m	100 m
Nominal TX power	0 10 dBm	-41.3 dBm/MHz	(25) 0 dBm	15 20 dBm
Number of RF channels	79	(1 15)	1/10 16	14 (2 4 GHz)
Channel bandwidth	1 MHz	500 MHz 7 5 GHz	0 3/0 6 MHz; 2 MHz	22 MHz
Modulation type	GFSK	BPSK QPSK	BPSK (ASK) O-QPSK	BPSK, QPSK COFDM CCK M-QAM
Spreading	FHSS	DS-UWB MB-OFDM	DSSS	DSSS CCK, OFDM
Coexistence mechanism	Adaptive freq hopping	Adaptive freq hopping	Dynamic freq selection	Dynamic freq selection, transmit power control (802 11h)
Basic cell	Piconet	Piconet	Star	BSS
Extension of the basic cell	Scatternet	Peer-to-peer	Cluster tree Mesh	ESS
Max number of cell nodes	8	8	65000	2007
Encryption	E0 stream cipher	AES block cipher (CTR, ounter mod)	AES block cipher (CTR counter mode)	RC4 stream cipher (WEP) AES block cipher
Authentication	Shared secret	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802 11)
Data protection	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

Unapproved draft.
Acronyms: ASK (amplitude shift keying), GFSK (Gaussian frequency SK), BPSK/QPSK (binary/quadrature phase SK), O-QPSK (offset-QPSK), OFDM (orthogonal frequency division multiplexing), COFDM (coded OFDM) MB-OFDM (multiband OFDM) M-QAM (M-ary quadrature amplitude modulation), CCK (complementary code keying), FHSS/DSSS (frequency hopping/direct sequence spread spectrum), BSS/ESS (basic/extended service set), AES (advanced encryption standard), WEP (wired equivalent privacy), WPA (Wi-Fi protected access) CBC-MAC (cipher block chaining message authentication code), CCM (CTR with CBC-MAC), CRC (cyclic redundancy check)

2 3 1 Tipos de Dispositivos ZigBee

Los nodos que forman parte de una red ZigBee se clasifican en 3 tipos

Coordinador Es el nodo más importante de la red ya que es el encargado de crearla Por este motivo solo puede existir uno por red Además se encarga de asignar direcciones a los nodos que se van uniendo a la red manteniendo la seguridad administrando el sistema y actuando de enlace con otras redes u otros dispositivos

Router Es un nodo ZigBee plenamente operativo Puede unirse a redes existentes transmitir recibir y encaminar información Actua como intermediario entre nodos que no pueden comunicarse directamente entre si ampliando el alcance efectivo de la red Además puede permitir a otros nodos el ingreso en la red Su consumo es mayor que el de los terminales puesto que al tener responsabilidades de enrutamiento no puede desconectarse en ningun momento

Terminal Es un nodo provisto con la funcionalidad básica para unirse a la red enviar y recibir información No puede actuar como intermediario entre otros nodos Necesita que un router o el coordinador sea su nodo padre y le permita el acceso a la red Puede entrar de forma cíclica en un modo de bajo consumo para ahorrar energía (Digi International (2008))

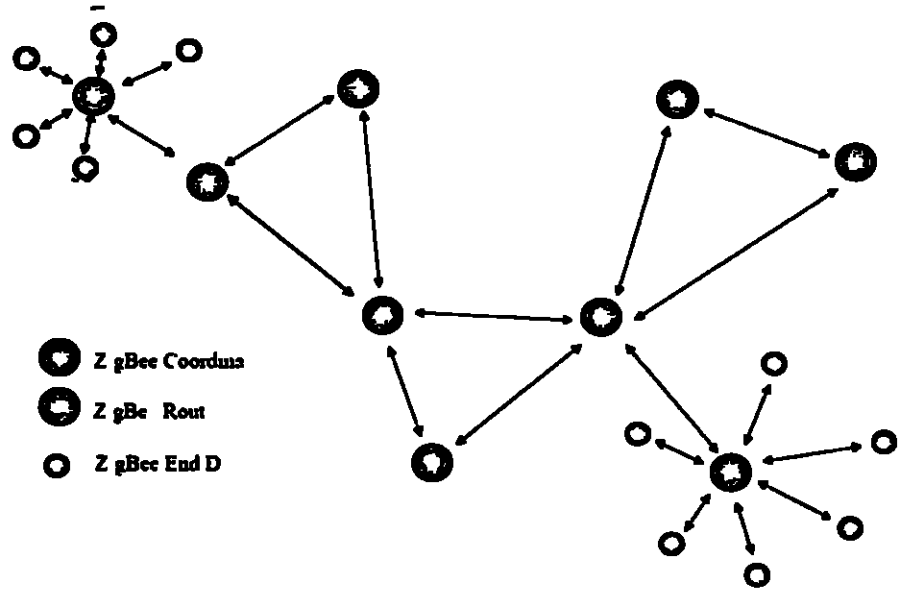


Fig 9 Topologia de una red ZigBee (Digi International (2008))

2.3 2 Direccionamiento

Cada dispositivo ZigBee posee una dirección unica, exclusiva de 64 bits denominada dirección MAC que le distingue de cualquier otro dispositivo a nivel mundial Adicionalmente también dispone de una dirección de red de 16 bits que le es asignada dinámicamente por el coordinador al asociarse a la red Dentro de una PAN (Personal Area Network) cualquier dispositivo puede ser identificado unívocamente por cualquiera de estas dos direcciones También se le puede asignar arbitrariamente un nombre identificativo formado por una cadena de caracteres pero este nombre no garantiza su identificación inequivoca dentro de una red

De igual modo cada red ZigBee posee dos identificadores que permiten distinguirla de cualquier otra red de su entorno incluso de las que operan en su mismo canal permitiendo su coexistencia en un mismo espacio El primero de estos identificadores se

denomina PAN ID y es un valor de 16 bits que se establece al crearse la red El segundo es el llamado Extended PAN ID un valor de 64 bits asignado arbitrariamente por el administrador de la red y que es añadido junto al PAN ID en los paquetes beacon de señalización para permitir que los nodos que deseen asociarse puedan identificar de forma unívoca la red

Dos redes ZigBee que compartan un mismo espacio se pueden diferenciar segun la frecuencia del canal sobre el que trabajan Cuando el nodo coordinador crea la red sondea los canales disponibles en la banda utilizada 2 4 GHz y elige el más adecuado Todos los nodos que se asocien a la red deben usar este mismo canal Por defecto los dispositivos ZigBee se encargan automáticamente de seleccionar el canal adecuado

En la (Fig 10) podemos observar tres redes ZigBee coexistiendo en un mismo espacio gracias a que pueden ser diferenciadas unívocamente por su PAN ID y el canal de transmisión De igual modo los nodos que la constituyen pueden tener la misma dirección de red que los nodos de las otras redes puesto que es un valor local a la PAN (Personal Area Network) (Digi International (2008))

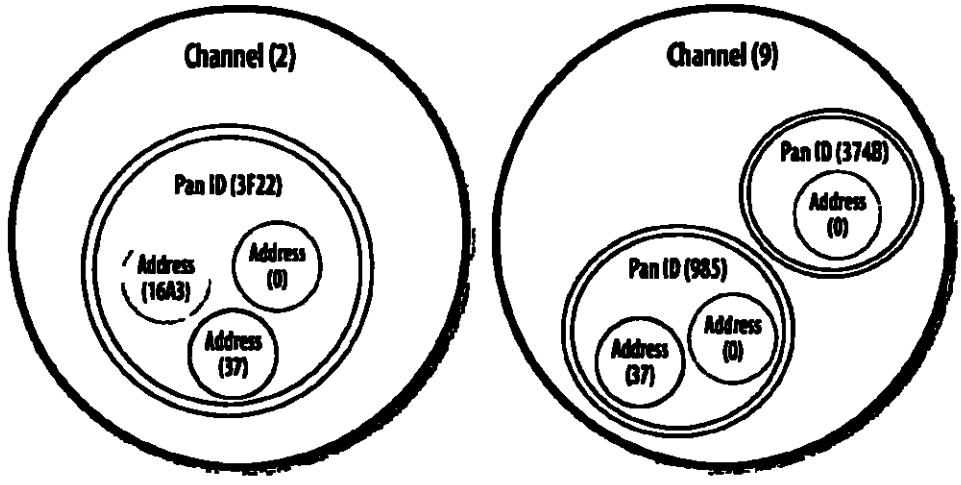


Fig 10 Coexistencia de redes ZigBee (Digi International (2008))

2.3 3 Seguridad

La tecnología ZigBee garantiza la seguridad de las comunicaciones mediante el mecanismo Advanced Encryption Standard o AES Se trata de un fiable rápido y eficiente algoritmo de cifrado en bloque que utiliza claves de 128 bits y que proporciona confiabilidad de datos integridad en las conexiones autenticación de los elementos del sistema y protección a la red frente a ataques malintencionados

ZigBee soporta dos modos de manejo de claves

Claves de red Proporcionan seguridad a las comunicaciones entre nodos Cada paquete es cifrado y enviado al siguiente nodo de la red donde es descifrado cifrado nuevamente y retransmitido al siguiente nodo en ruta hasta su destino

Claves de enlace Proporcionan seguridad extremo a extremo El origen del paquete lo cifra, y este permanece cifrado mientras se transmite por la red hasta que llega a su destino donde es descifrado Es conveniente usar este tipo de claves en situaciones donde algunos de los nodos o enlaces intermedios de la red no son completamente confiables (Digi International (2008))

2 4 XBee

Se trata de una gama de módulos de radio fabricados por la empresa llamada Digi International con una amplia variedad de modelos componentes firmware potencias de transmisión y antenas Trabajan con un voltaje de operación de 3 3 voltios y aunque incorporan hasta 20 pines para distintas operaciones como reinicio o modo de bajo

consumo pueden funcionar haciendo uso tan solo de sus 4 pines principales alimentación tierra entrada y salida de datos

Cuenta con un microchip que además de ocuparse de las tareas propias de la comunicación radio permite realizar operaciones con un nivel de lógica muy básico como por ejemplo leer directamente datos de sensores y retransmitirlos sin tener que contar con procesado externo Para lógica más avanzada deben conectarse a otro dispositivo que se encargue de ello como un micro-controlador más complejo (Oyarce (2010))

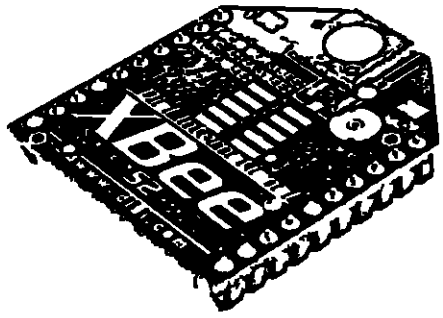


Fig 11 Módulo XBee Serie 2

Los modulos XBee se clasifican principalmente en dos tipos

Serie 1 Esta serie de módulos XBee proporciona soporte para comunicaciones punto a punto de forma simple y basándose en el estándar 802 15 4 Su principal ventaja es la sencillez

Serie 2 Estos módulos permiten implementar redes malladas siguiendo el protocolo ZigBee Además poseen un mayor alcance y un menor consumo que la serie 1

Los módulos XBee de la serie 1 y la serie 2 no son compatibles entre sí y no pueden interactuar entre ellos de ninguna forma Los que pertenecen a una serie solo se pueden comunicar con otros de la misma serie

Cuadro 2 XBee comparativa de series

	XBee Series 1	XBee Series 2
Indoor/Urban range	up to 100 ft (30m)	up to 133 ft (40m)
Outdoor RF line of sight range	up to 300 ft (100m)	up to 400 ft (120m)
Transmit Power Output	1 mW (0dbm)	2 mW (+3dbm)
RF Data Rate	250 Kbps	250 Kbps
Receiver Sensitivity	92dbm (1% PER)	98dbm (1% PER)
Supply Voltage	2 8 3 4 V	2 8 3 6 V
Transmit Current (typical)	45 mA (@ 3 3 V)	40 mA (@ 3 3 V)
Idle/Receive Current (typical)	50 mA (@ 3 3 V)	40 mA (@ 3 3 V)
Power down Current	10 uA	1 uA
Frequency	ISM 2 4 GHz	ISM 2 4 GHz
Dimensions	0 0960 x 1 087	0 0960 x 1 087
Operating Temperature	40 to 85 C	40 to 85 C
Antenna Options	PCB Integrated Whip U FL RPSMA	PCB Integrated Whip U FL RPSMA
Network Topologies	Point to point Star Mesh (with DigiMesh firmware)	Point to point Star Mesh
Number of Channels	16 Direct Sequence Channels	16 Direct Sequence Channels
Filtration Options	PAN ID Channel & Source/Destination	PAN ID Channel & Source/Destination

2 4 1 Modos de Configuración de los Modulos XBee

Las radios XBee presentan dos modos de funcionamiento denominados modos AT y API

2 4 1 1 Modo AT

Se trata de un modo transparente donde la información recibida por el pin de entrada de datos es retransmitida por radio sin ninguna modificación Los datos pueden ser

enviados a un unico destinatario (punto a punto) o bien pueden ser transmitidos a multiples destinos (broadcast)

2 4 1 2 Modo API

Este modo es más complejo pero permite el uso de frames con cabeceras que aseguran la entrega de los datos al estilo TCP Extiende el nivel en el cual la aplicación del cliente puede interactuar con las capacidades de la red del módulo Cuando el módulo XBee se encuentra en este modo toda la información que entra y sale es empaquetada en frames que definen operaciones y eventos dentro del módulo Asi un Frame de Transmisión de Información incluye

Frame de información RF transmitida

Frame de comandos (equivalente a comandos AT)

Mientras que un Frame de Recepción de Información incluye

Frame de información RF recibida

Comando de respuesta

Notificaciones de eventos como Reset Associate Disassociate etc

Esta API provee alternativas para la configuración del módulo y ruteo de la información en la capa de aplicación del cliente Un cliente puede enviar información al módulo XBee Estos datos serán contenidos en un frame cuya cabecera tendrá información util referente del módulo Esta información además se podrá configurar es decir que en vez de estar usando el modo de comandos para modificar las direcciones la API lo realiza automáticamente El módulo asi enviará paquetes de datos contenidos en frames a otros módulos de destino con información a sus respectivas aplicaciones

conteniendo paquetes de estado asi como el origen RSSI (potencia de la señal de recepción) e información de la carga util de los paquetes recibidos

Entre las opciones que permite la API se tienen

Transmitir información a multiples destinatarios sin entrar al modo de Comandos

Recibir estado de éxito/falla de cada paquete RF transmitido

Identificar la dirección de origen de cada paquete recibido

El modo API se divide a su vez de dos modos AP = 1 y AP = 2 Con AP = 1 el módulo trabaja en el modo API Y con AP = 2 el módulo trabaja en modo API pero con Carácter de Escape Este modo es necesario sólo cuando se envian bytes que interfieren con la estructura del Frame Este modo ingresa un carácter de escape además de otra operación sobre el bytes de interferencia Esto hace que el frame sea mas grande al agregar más bytes pero evita que la cabecera del frame se confunda con los datos enviados Otra ventaja es el Checksum que permite verificar que los datos entregados no se hayan corrompido

Entre las posibilidades que permite el modo API esta cambiar parámetros a través de comandos AT enviándolos al módulo de destino Asi desde un módulo es posible configurar otro También es posible consultar sobre el estado de algun parámetro en otro módulo Además se puede consultar sobre el estado del módulo como saber si está asociado a un coordinador o si es o no un coordinador

2 4 2 Direccionamiento de los Modulos XBee

Permiten dos tipos de direccionamiento los cuales son de 16 bit y 64 bits La principal diferencia es que en el de 64 bit es posible obtener una mayor cantidad de direcciones y

por lo tanto una mayor cantidad de nodos o equipos funcionando en la misma red Son a través de estas direcciones que los módulos se comunican entre si

Direccionamiento de 16 bits Al configurar el modulo con este tipo de direccionamiento podemos designarle cualquier dirección de 16 bits para de esta forma identificarlo dentro de nuestra red

Direccionamiento de 64 bits Con este direccionamiento ya no es posible definir la dirección de origen del módulo ya que ésta se asigna automáticamente En este caso la dirección del módulo corresponde a su numero serial que viene de fábrica y el cual es imposible de cambiar Este numero se encuentra guardado en dos variables de 32 bit cada una, llamadas SL (Source Low) y SH (Source High) y es unico SL lee los 32 bit menos significativos del numero serial y SH los 32 más significativos

2 4 3 Trama del Modo API

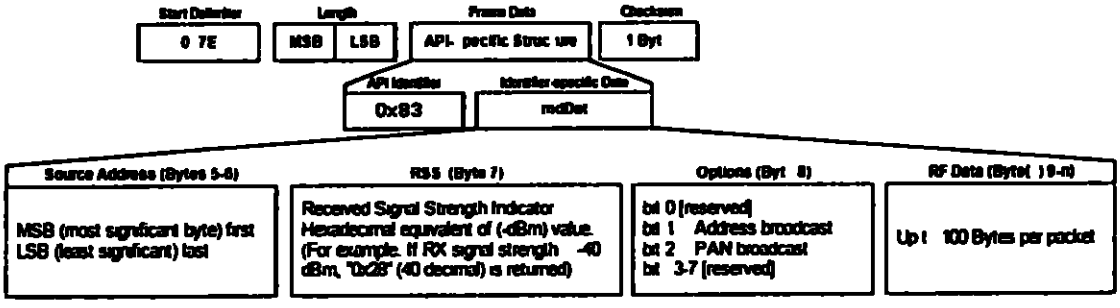


Fig 12 Trama del modo API (Digi International (2008))

El primer byte de la trama 0x7E indica el comienzo del frame Los dos bytes siguientes indican el largo solamente del Frame de Datos (Frame Data) es decir todo el frame sin contar ni el byte 0x7E ni el largo (Length) ni el byte (Checksum) La

estructura API se compone del identificador API 0x83 (cuarto byte del frame) indica que se están recibiendo datos utilizando direccionamiento de 16 bits Los dos bytes siguientes (el byte 5 y 6) indican la dirección de origen El byte 7 indica el RSSI (potencia de la señal de recepción) que permite determinar la potencia de la señal desde donde vienen los datos El byte 8 se divide en dos bits de los cuales el bit 1 indica si es un broadcast de direccionamiento utilizando la misma PAN y el bit 2 que indica un broadcast de todas las redes PAN Luego a partir del byte 9 hasta el numero de byte dado por el byte de Length corresponden a los datos obtenidos del modo Cable Virtual provenientes de otro módulo

2 4 4 Software X CTU

Para programar los módulos XBee se debe cargar el firmware deseado y modificar el valor de los distintos parámetros que determinan su comportamiento Para ello se utiliza un software distribuido por la empresa Digi International llamado X CTU el mismo se puede descargar sin costo desde la página web del fabricante (www digi com)

A pesar de que la unica forma de actualizar el firmware de un módulo XBee es mediante el software X CTU para modificar otras configuraciones se puede hacer uso de cualquier otro programa terminal serie entrar en modo comando y reconfigurar el modulo mediante el envió de comandos AT

Otra forma de configurar y actualizar el firmware de los módulos XBee es a través del micro-controlador Arduino para ello se debe crear un sketch en blanco dentro del entorno Arduino luego se conecta el módulo XBee a nuestro micro controlador y por

ultimo a nuestra PC De esta forma el Arduino funciona como puente entre el XBee y la PC

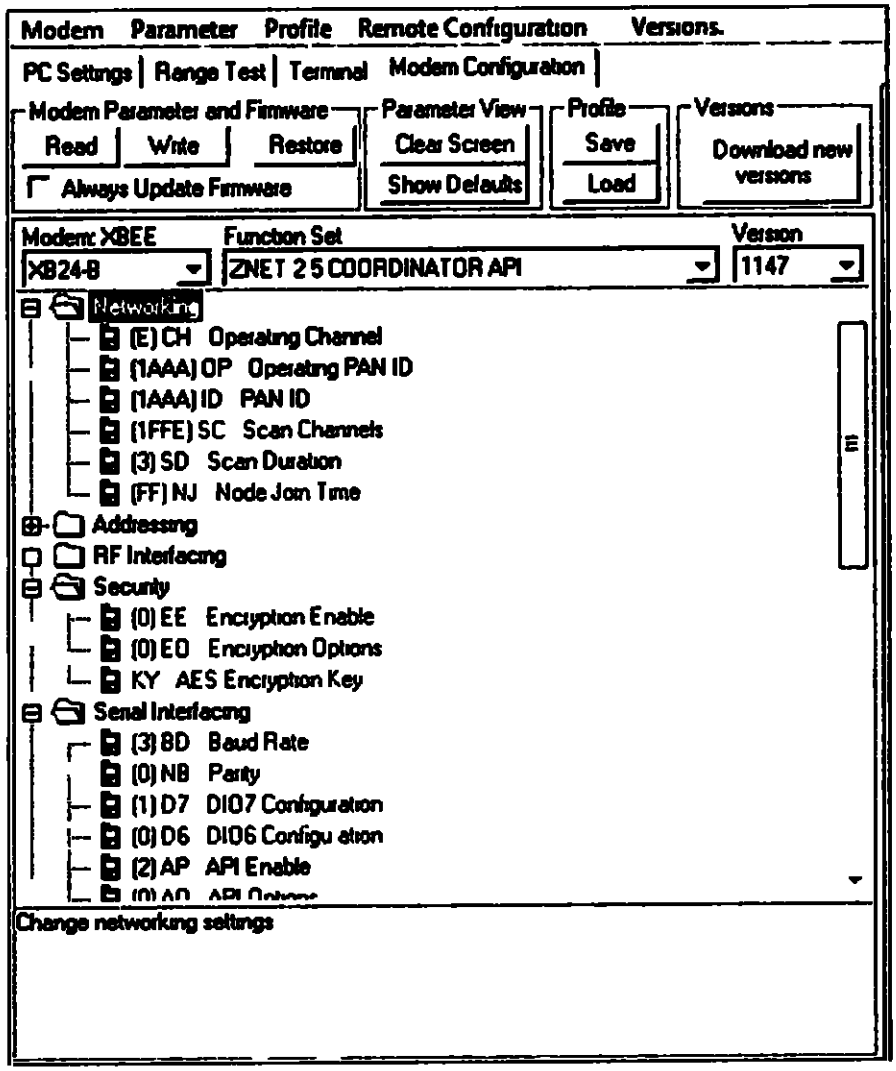


Fig 13 Software X-CTU

2 5 Arduino

Es una placa de hardware y software libre desarrollada para la creación de prototipos de forma flexible y fácil de utilizar Se creo para diseñadores aficionados profesionales

investigadores y para cualquiera interesado en crear alguna solución en nuestro entorno real por medio de la electrónica.

El diseño del hardware está inspirado originalmente en el hardware libre preexistente llamado Wiring y el software del entorno de desarrollo en la guía del lenguaje de programación libre llamado Processing. A pesar de que el entorno Processing está construido internamente con código escrito en lenguaje Java, los creadores de Arduino se basaron en el lenguaje de código llamado C/C++.

En el mercado actual existe una gran variedad de modelos de placas Arduino, cada una con diferentes características, como por ejemplo, el tamaño físico, el número de pines hembra ofrecidos, el modelo del micro-controlador incorporado, la cantidad de memoria utilizada, etc. Aunque puedan ser modelos específicos diferentes, los micro-controladores incorporados en las diferentes placas pertenecen todos a la misma familia tecnológica, por lo que su funcionamiento en realidad es bastante parecido entre sí. En concreto, todos los micro-controladores son de tipo AVR, una arquitectura desarrollada y fabricada por la marca Atmel. Por ello es tan importante observar y analizar las diferentes características de cada modelo para elegir la más adecuada para nuestro desarrollo.

Los proyectos desarrollados en Arduino pueden ser autónomos o no, ya que, una vez programado su micro-controlador la placa no necesita estar conectada a ninguna computadora y puede funcionar de forma autónoma si dispone de una fuente de alimentación.

Arduino puede tomar información del entorno mediante una gama de sensores a través de sus pines de entrada y por medio de sus terminales de salida controlar aquello que le rodea, como por ejemplo luces, motores y otros actuadores. Gracias a lo anterior, se

pueden realizar multitud de proyectos de rango muy variado robótica domótica monitorización de sensores ambientales sistemas de navegación telemática etc pensando ello lo unico que nos limita son nuestros conocimientos en electrónica y programación

Decimos que el entorno de desarrollo es de software libre porque se publica con una combinación de licencias GPL (Licencia publica general) para el entorno visual de programación y la licencia LGPL (Licencia publica general reducida) para los códigos fuente de gestión y control del micro-controlador a nivel más interno La conclusión de lo anterior es que cualquier persona que quiera y sepa, puede formar parte del desarrollo del software Arduino y contribuir asi a mejorar dicho software aportando nuevas caracteristicas sugiriendo ideas de nuevas funcionalidades compartiendo soluciones a posibles errores existentes etc

Arduino se califica como hardware libre (open source) porque sus ficheros esquemáticos están disponibles para descargar de la página web del proyecto con la licencia *Creative Commons Attribution Share Alike* la cual es una licencia libre que permite realizar trabajos derivados tanto personales como comerciales siempre que estos den crédito a Arduino y publiquen sus diseños bajo la misma licencia Con lo anterior concluimos que podemos realizar los cambios que queramos al esquema físico de Arduino y confeccionar nuestra propia placa personalizada

2 5 1 Características Comunes en las Placas Arduino

A continuación explicamos una serie de características que independientemente del modelo de la placa Arduino las mismas siempre están presentes

2 5 1 1 Memoria Flash

Es una memoria persistente donde se almacena permanentemente el programa que ejecuta el micro-controlador hasta una nueva reescritura si se da el caso

En los micro-controladores que vienen incluidos en la placa Arduino no podemos utilizar toda la capacidad de la memoria Flash porque existen 512 bytes utilizados por el llamado *bootloader block*' (gestor de arranque) el mismo viene ya pre programado de fábrica, el cual nos permite usar la placa Arduino de una forma sencilla y cómoda sin tener que conocer las interioridades electrónicas más avanzadas del micro-controlador

2 5 1 2 Memoria Estática de Acceso Aleatorio

Memoria volátil donde se alojan los datos que en ese instante el programa necesita crear o manipular para su correcto funcionamiento es decir las variables utilizadas en el programa creado por nosotros Si necesitáramos ampliar la cantidad de memoria SRAM (Memoria Estática de Acceso Aleatorio) disponible siempre podríamos adquirir memorias SRAM independientes y conectarlas al micro-controlador utilizando algun protocolo de comunicación conocido por este como SPI o I^2C

2 5 1.3 Memoria de solo lectura programable y borrada electricamente

La memoria EEPROM (Memoria de solo lectura programable y borrada electricamente) es de tipo persistente donde se almacenan los datos que deseamos queden grabados una vez apagado el micro-controlador para poderlos utilizar posteriormente en siguientes ejecuciones Si necesitáramos ampliar la cantidad de memoria EEPROM

disponible siempre podríamos adquirir memorias EEPROM independientes y conectarlas al micro-controlador utilizando algun protocolo de comunicación conocido por este como SPI o I^2C Alternativamente adquirir tarjetas de memoria de tipo SD (Secure Digital) y comunicarlas mediante un circuito especifico al micro-controlador

Analizando la forma de trabajar de las memorias antes mencionadas podemos deducir que la arquitectura a la que pertenece en general toda la familia de micro-controladores AVR es de tipo Harvard ya que la memoria que aloja los datos (SRAM o EEPROM) está separada de la memoria que aloja las instrucciones (Flash) por lo que ambas memorias se comunican con la CPU de forma totalmente independiente y en paralelo consiguiendo así una mayor velocidad y optimización La arquitectura Von Neumann es la que comunmente vemos en las PC s de hoy dia, en la cual la CPU está conectada a una memoria RAM unica que contiene tanto las instrucciones del programa como los datos por lo que la velocidad de operación está limitada por el efecto cuello de botella que significa un unico canal de comunicaciones para datos e instrucciones (Torres (2013))

2 5 1 4 Inter Integrated Circuit

I^2C es un protocolo de transmisión serial muy utilizado en la industria principalmente para comunicar circuitos integrados entre si Su principal característica es que utiliza dos lineas para transmitir la información una llamada linea SDA que sirve para transmitir los datos los 0s y los 1s y otra llamada linea SCL utilizada para enviar la señal de reloj Además necesitaremos dos lineas más una para la alimentación y otra para la tierra La velocidad de transferencia de datos es de 100Kbits por segundo en el modo estándar aunque también se permite velocidades de hasta 3 4 Mbits por segundo La transmisión

de la información es half duplex es decir la comunicación solo se puede establecer en un sentido al mismo tiempo por lo que en el momento que un dispositivo empiece a recibir un mensaje tendrá que esperar a que el emisor deje de transmitir para poder responder (Torres (2013))

2 5 1 5 Serial Peripheral Interface

SPI es un estándar que permite controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie sincronizado Requiere de cuatro cables uno llamado SCK que envía a todos los dispositivos la señal de reloj generada por el maestro actual otro llamado SS es el utilizado por el maestro para elegir en cada momento con que dispositivo esclavo se quiere comunicar de entre los varios que puedan estar conectados ya que solo puede transferir datos con un solo esclavo a la vez MOSI para enviar los datos 0s y 1s desde el maestro hacia el esclavo elegido y por ultimo MISO que envia los datos en sentido contrario del esclavo al maestro Analizando lo anterior al haber dos cables para los datos la transmisión es full duplex es decir que la información puede ser transmitida en ambos sentidos al mismo tiempo (Torres (2013))

2 5 1 6 La Alimentación

El voltaje de alimentación de la placa Arduino es de 5V Podemos obtener esta alimentación de diferentes formas

Conectando la placa Arduino a una fuente externa como por ejemplo un adaptador AC/DC o una pila Si es un adaptador la placa dispone de un zócalo donde se puede enchufar una clavija de 2 1mm de tipo *jack* Utilizando una pila necesitamos

conectar el polo positivo al pin con nombre *Vin* y el negativo al pin *Gnd* en la zona de la placa marcada con la etiqueta *POWER* En ambos casos la placa está preparada en teoria para recibir una alimentación de 6 a 20 voltios aunque realmente el rango recomendado de voltaje de entrada es de 7 a 12 voltios En cualquier caso este voltaje de entrada ofrecido por la fuente externa siempre es rebajado a los 5V de trabajo mediante un circuito regulador de tensión que ya viene incorporado dentro de la placa

Conectando la placa Arduino a nuestro computador mediante un cable USB La placa dispone de un conector USB hembra de tipo B La alimentación recibida de esta manera está regulada permanentemente a los 5V de trabajo y ofrece un máximo de hasta 500mA de corriente por lo tanto la potencia consumida por la placa es en ese caso de unos 2 5W Si en algun momento por el conector USB pasa más intensidad de la deseable la placa Arduino está protegida mediante un poli fusible que automáticamente rompe la conexión hasta que las condiciones eléctricas vuelvan a la normalidad

2 5 1 7 Las Entradas y Salidas Digitales

Es aqui donde conectamos nuestros sensores para que la placa pueda recibir datos del entorno y también donde conectaremos los actuadores para que la placa pueda enviarles las ordenes pertinentes además de poder conectar cualquier otro componente que necesite comunicarse con la placa de alguna manera Todos estos pines digitales funcionan a 5V pueden proveer o recibir un máximo de 40mA y disponen de una resistencia pull up interna de entre 20KΩ y 50KΩ que inicialmente está conectada salvo que nosotros indiquemos lo contrario mediante programación software Hay que tener en cuenta que aunque cada pin individual pueda proporcionar hasta 40mA como

máximo en realidad internamente la placa agrupa los pines digitales de tal forma que se pueda brindar un máximo de 100mA en total dividido en diferentes pines trabajando a la vez

2 5 1 8 Las Entradas Analógicas

Pueden recibir voltajes dentro de un rango de valores continuos de entre 0 y 5V La electrónica de la placa solo puede trabajar con valores digitales por lo que es necesario una conversión previa del valor analógico recibido a un valor digital lo más aproximado posible Esta se realiza mediante un circuito conversor analógico/digital incorporado en la propia placa Cada canal del circuito conversor dispone de 10 bits los llamados bits de resolución para guardar el valor del voltaje convertido digitalmente Analizando lo anterior podemos decir que la placa Arduino puede representar 2^{10} (1024) combinaciones diferentes de 0s y 1s Si dividimos el voltaje máximo de entrada 5V entre el numero de combinaciones (1024) obtendremos que cada valor digital corresponde aproximadamente a 5mV En otras palabras todos los valores analógicos dentro de cada rango de 5mV desde 0 a 5V se unen sin distinción en un unico valor digital desde 0 a 1023 Por este motivo no podremos distinguir valores analógicos distanciados por menos de 5mV

Una funcionalidad extra que podemos utilizar a través de estos pines es utilizarlos como entradas salidas digitales siendo estos habilitados mediante código de programación

2 5 1 9 Salidas Analógicas por Modulación de Ancho de Pulsos

La placa Arduino no dispone de salidas analógicas si no que utiliza algunos pines de salidas digitales para simular un comportamiento analógico los mismos etiquetados con *PWM* (Pulse Width Modulation modulación por ancho de pulso) Lo que hace este tipo de señal es emitir en lugar de una señal continua una señal cuadrada formada por pulsos de frecuencia constante aproximadamente de 490Hz Al variar la duracion de estos pulsos estaremos variando proporcionalmente la tensión promedio resultante cuanto más cortos sean los pulsos menor será la tensión promedio de salida y cuanto más largos sean los pulsos mayor será dicha tensión Cada pin PWM tiene una resolución de 8 bits por lo tanto podemos representar una combinación de 2^8 (256) valores diferentes posibles teniendo disponibles 256 valores diferentes para indicar la duración deseada de los pulsos de la señal para de esta forma obtener mediante una duración de pulso de (0) una señal estrecha y tener entonces un valor de salida de 0V pero si utilizamos una duración de pulso de (255) obtendremos una señal de duración máxima generando una salida de 5V Para obtener diferencia de salida de voltaje con cada duracion de pulso tendremos que dividir la salida máxima (5V) entre el numero de resolución 2^8 (256) esto nos da 19 5mV es decir cada valor promedio esta distanciado del anterior y del siguiente por una diferencia de 19 5mV

2 5 1 10 Pin 0 (RX) y Pin 1 (TX)

Permite que el micro controlador pueda recibir directamente datos en serie por el pin RX o transmitirlos por el pin TX sin pasar por la conversión USB Serie De todas formas

estos pines están internamente conectados mediante resistencias de 1 KΩ a la conexión USB Serie por lo que los datos disponibles en el USB también lo estarán en estos pines

2 5 1 11 Pin 13

Este pin está conectado directamente a un LED (diodo emisor de luz) incrustado en la placa identificado con la etiqueta L de forma que si el valor del voltaje recibido por este pin es *ALTO* (HIGH) el LED se encenderá y si es *BAJO* (LOW) el LED se apagara

2 5 1 12 Pin AREF

Ofrece un voltaje de referencia externo para poder aumentar la precisión de las entradas analógicas

2 5 1 13 Pin RESET

Si el voltaje de este pin se establece a valor *BAJO* (LOW) el micro-controlador se reiniciara y se pondrá en marcha el bootloader Ofrece la posibilidad de añadir otro botón de reinicio a placas supletorias (Shields Placas que se conectan al Arduino para ampliar sus prestaciones) las cuales por su colocación puedan ocultar o bloquear el botón de la placa Arduino

2 5 1 14 Pin IOREF

Este pin es una duplicación regulada del pin *Vin* su función es indicar a las placas supletorias conectadas a nuestra placa Arduino el voltaje al que trabajan los pines de

entrada/salida de esta, para que las placas supletorias se adapten automáticamente a ese voltaje de trabajo

2 5 1 15 El Botón de Reset

Permite una vez pulsado enviar una señal LOW al pin *RESET* de la placa para apagar y volver a arrancar el micro-controlador

2 5 2 Arduino Uno

Es el modelo de referencia de la plataforma Arduino alcanzando un compromiso entre la sencillez de los modelos más básicos y la potencia de los más complejos Está basado en el micro-controlador ATmega328 Dispone de 14 pines de entrada/salida digital (de los cuales 6 pueden ser utilizados como salida PWM y 2 como puerto serie) 6 entradas analógicas reloj cerámico de 16 MHz puerto USB conector de alimentación cabezal para programación en circuito serie y un botón de reinicio entre otros componentes

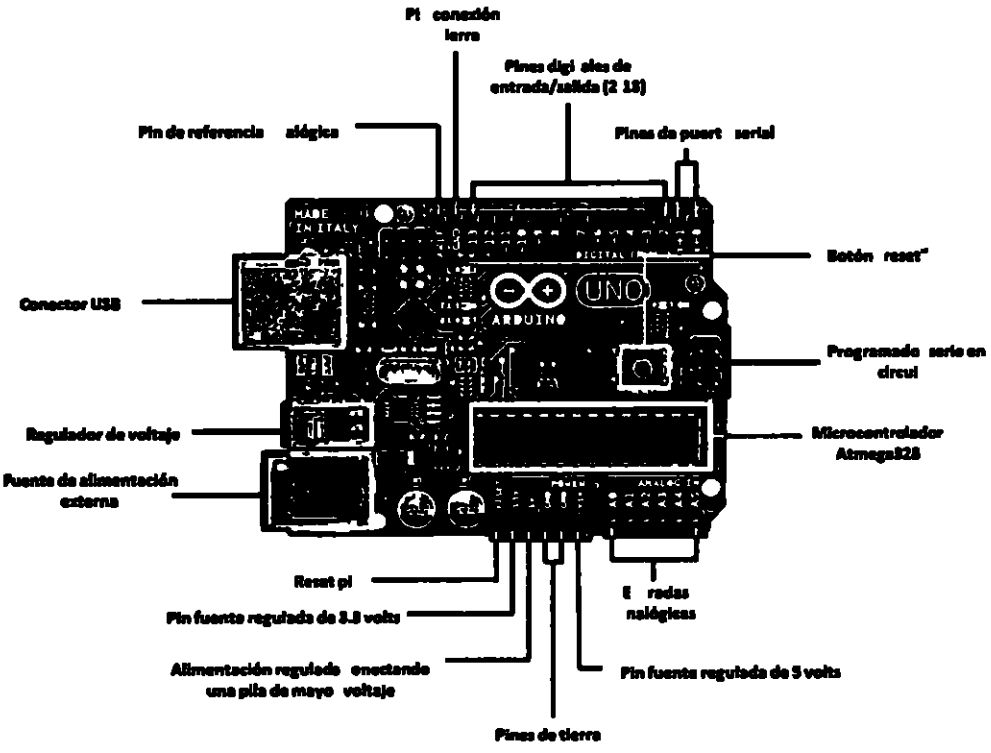


Fig 14 Partes de la placa Arduino Uno

A continuación en el (Cuadro 3) se muestra las características de la placa Arduino Uno

Cuadro 3 Características técnicas del Arduino Uno

[illegible]	[illegible]
Operating Voltage	5V
Input Voltage (recommended)	7 12V
Input Voltage (limits)	6 20V
Digital I/O Pins	14 *(of which 6 provide PWM output)*
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3 3V Pin	50 mA
Flash Memory	32 KB (ATmega 328) *(of which 0 5 KB used by bootloader)*
SRAM	2 KB (ATmega328)
EFPROM	1 KB (ATmega328)
Clock Speed	16 MHz

2 5.3 Arduino Mega Accessory Development Kit

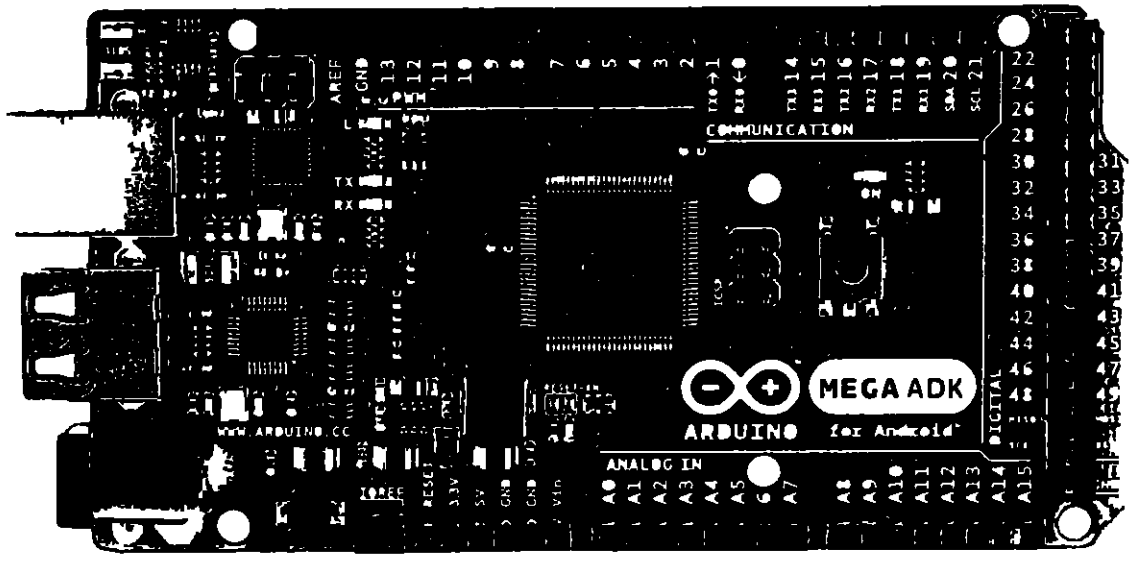


Fig 15 Apariencia del Arduino Mega ADK

Lo destacable de esta placa es su capacidad de funcionar como un dispositivo de tipo host USB (anfitrión USB) En una comunicación USB entre diferentes dispositivos

siempre existe uno que actua como maestro al que se le llama *host* y el otro u otros que actuan como esclavos llamados *periféricos* El host es el unico que puede iniciar y controlar la transferencia de datos entre el resto de dispositivos conectados mientras que los periféricos tan solo pueden responder a las peticiones hechas por el host Los dispositivos host disponen de un conector USB de tipo A y los dispositivos periféricos disponen de un conector USB de tipo B mini B o micro B Un host tipico es un computador al cual se le pueden conectar varios periféricos como memorias USB tipo *Pen drive* cámaras fotográficas cámaras de video Smartphone etc

El Arduino Mega ADK puede funcionar como un periférico USB normal como el resto de las placas Arduino pero además como un host USB El modo de funcionamiento como host USB es gracias a que la placa incorpora un micro-controlador especifico que implementa la lógica necesaria para realizar este trabajo llamado MAX3421E del fabricante Maxim el cual se comunica con el ATmega2560 De esta forma se le puede conectar cualquier dispositivo que tenga un puerto USB periferico para controlarlo e interactuar directamente con él

Mega ADK esta especificamente diseñado para trabajar directamente con Smartphone funcionando con el sistema operativo Android desarrollado por Google La idea es que se puedan escribir programas para Android que se relacionen con el código Arduino ejecutado en la placa, de tal forma que se establezca una comunicación entre ambos como por ejemplo realizar un control remoto desde el dispositivo Android de los sensores y actuadores al hardware Arduino

Cuenta con el micro controlador ATmega2560 integra 54 pines de entrada/salida digitales de las cuales 14 pueden ser utilizadas como salidas analógicas PWM además 16

entradas analógicas y 4 pares de receptores/transmisores de tipo serie TTL UART (Transmisor/receptor asincrono universal) A continuación en el (Cuadro 4) se muestran las caracteristicas técnicas del Arduino Mega ADK

Cuadro 4 Caracteristicas técnicas Arduino Mega ADK

Microcontrolador	ATmega2560
Alimentacion	5V
Entrada	7 12V
Limites (max)	5 5 16V
Pines digitales	54 (14 con PWM)
Pines analogicos	16
Corriente por pin	40 mA
Corriente sobre pin 3 3V	50 mA
Memroia Flash (programa)	256 KB (8 KB usados para el bootloader)
SRAM	8 KB
EEPROM	4 KB
Reloj	16 MHz

2 5 4 Arduino Integrated Development Environment

Para programar el micro-controlador se proporciona el oficial Arduino IDE (Integrated Development Environment) que se puede descargar de forma gratuita desde la página oficial de la plataforma Arduino Este software es multiplataforma por lo tanto se puede utilizar en sistemas operativos Windows Mac o Linux y proporciona herramientas como editor de código compilador cargador y monitor serial además de librerias y ejemplos

El lenguaje de programación Arduino está basado en C/C++ Los programas denominados comunmente sketches se dividen en tres partes principales estructura, variables y funciones Además todo programa debe comprender dos funciones esenciales denominadas setup() y loop()

setup() Esta función se carga cuando se inicia un sketch Se emplea para iniciar variables establecer el estado de los pines inicializar librerias etc Se ejecutará una unica

vez después de que se conecte la placa Arduino a la fuente de alimentación o cuando se pulse el botón de reinicio de la placa

loop() Esta función se ejecuta inmediatamente después de setup() Se trata de un bucle infinito que se repite indefinidamente mientras la placa esté en funcionamiento y no sea reiniciada En ella escribiremos el código que describirá el comportamiento principal que deseamos tenga nuestra palca Arduino

En la (Fig 16) se muestra el ciclo de vida de un sketch Arduino

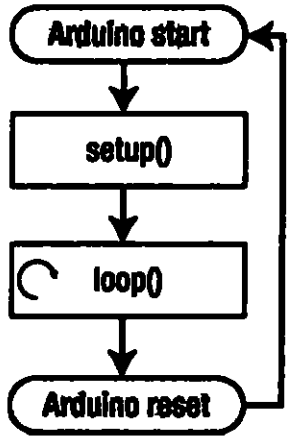


Fig 16 Ciclo de vida de un Sketch Arduino

A continuación en la (Fig 17) mostramos un vistazo al IDE Arduino y la estructura principal de un Sketck

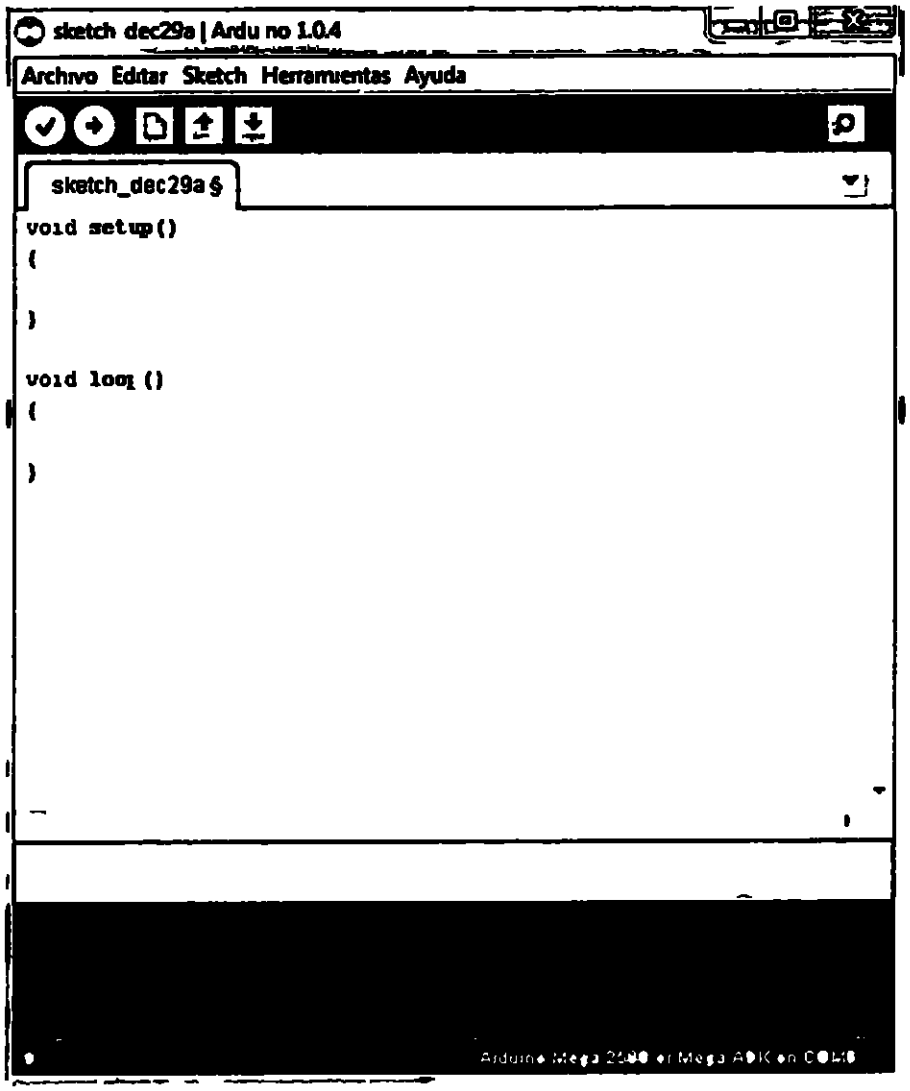


Fig 17 IDE Arduino

2 6 Sistema Operativo Android

Es un sistema operativo basado en Linux y distribuido por Google bajo licencia *Apache* gracias a lo anterior Android se califica como un sistema operativo de código abierto Fue diseñado principalmente para dispositivos móviles de pantalla táctil como Smartphone o tablets Su naturaleza de código abierto y su licencia de distribución hacen

posible que el software basado en Android sea libremente creado modificado y distribuido por cualquier desarrollador Los programas en entorno Android denominados comunmente aplicaciones o Apps se desarrollan en una versión personalizada del lenguaje de programación Java Google facilita completo acceso al kit de desarrollo de software o SDK (Software Development Kit) que proporciona todas las herramientas necesarias para implementar probar y depurar aplicaciones junto con prestaciones adicionales como librerias API documentación emulador códigos de ejemplos y tutoriales Además se permite a los desarrolladores la libre publicación de sus creaciones facilitando su distribución a través de la plataforma Google Play Actualmente se facilita un paquete en la página oficial de Google para desarrolladores (http //developer android com/sdk/index html) llamado ADT' (Android Development Tools) el mismo contiene el entorno de desarrollo *Eclipse* y el plugin correspondiente para desarrollar aplicaciones Android solo se necesita descargar descomprimir y ejecutar Eclipse para comenzar a trabajar en la plataforma Android Además recientemente ha sido publicado por Google un entorno de desarrollo especifico denominado Android Studio disponible de forma gratuita desde su web (http //developer android com/sdk/installing/studio html)

2 6 1 Niveles en la Estructura del Sistema Android

Está conformado por una estructura software dividida en diversos niveles Cada capa de esta arquitectura proporciona servicios a su capa inmediatamente superior

Nucleo Linux Android se construyó sobre un nucleo Linux personalizado por Google Este nivel se encarga de interactuar con el hardware proporcionando los

controladores necesarios Además actua como capa de enlace entre el hardware y el resto de capas software de la arquitectura

Librerías y Android Runtime Las librerias en este sistema operativo están escritas en código C/C++ Android Runtime consta de la máquina virtual Dalvik encargada de ejecutar las aplicaciones y un conjunto de librerias Java

Marco de Aplicación Este nivel también es llamado *framework* Gestiona funciones básicas del dispositivo y proporciona las herramientas sobre las que se construyen las aplicaciones

Aplicaciones Es el nivel superior en la estructura Android Incluye todas las aplicaciones desde las incorporadas por defecto en Android como las desarrolladas por terceros e instaladas por el usuario

En la (Fig 18) se muestran los niveles de la estructura Android

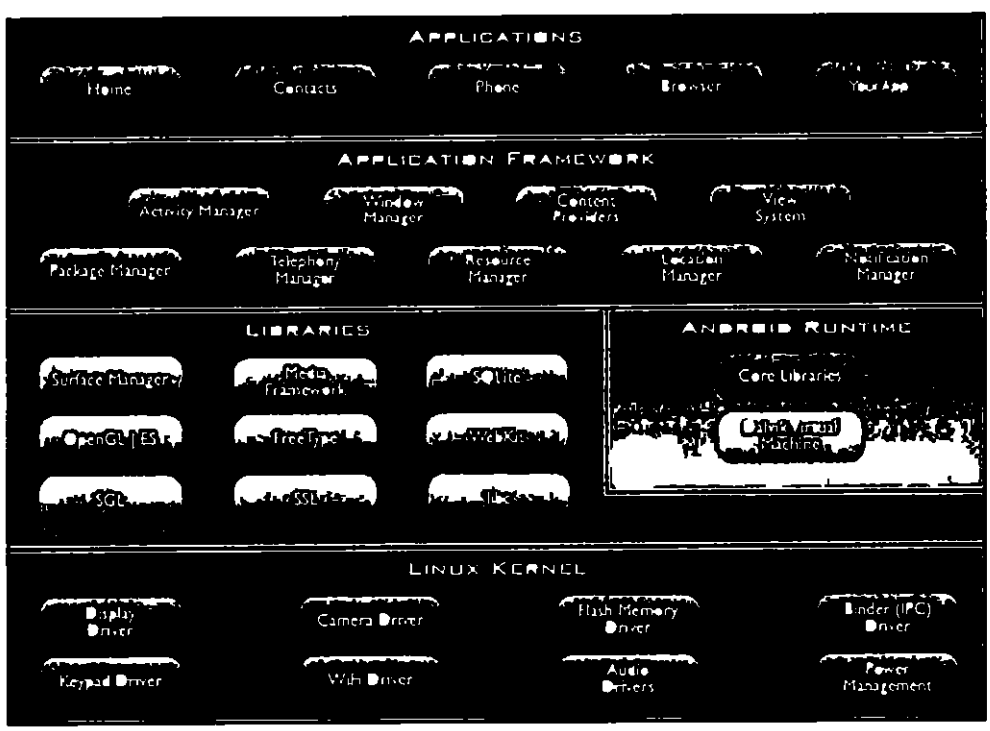


Fig 18 Niveles en la estructura Andorid

2 6 2 Actividades y Aplicaciones

Una actividad es un componente de una aplicación Android que proporciona una interfaz con la que el usuario puede interactuar Cada aplicación posee su propio proceso y en él se encuentran todas sus actividades Normalmente este proceso ocupa toda la pantalla del dispositivo aunque puede ser más pequeño o situarse por encima de otras actividades

Una aplicación puede constar de varias actividades vinculadas entre si Una de estas actividades es la principal que es presentada al usuario en el momento de lanzar la aplicación Cada actividad puede iniciar otras actividades para realizar distintas acciones Cuando se inicia una nueva actividad la anterior es detenida, pero preservada en una pila de actividades Se trata de una pila LIFO (last in first out) que permite volver a la actividad inmediatamente anterior pulsando el botón *back* del dispositivo

Durante la navegación del usuario por la aplicación las actividades transitan entre distintos estados en lo que se denomina el ciclo de vida de la actividad Cuando una actividad cambia de un estado a otro el sistema llama a determinados métodos que determinan su comportamiento al transitar a ese nuevo estado realizando las acciones correspondientes como por ejemplo iniciar variables en caso de crearse una nueva actividad o liberar recursos en caso de detenerse Estos métodos pueden ser modificados por los desarrolladores personalizando su comportamiento o incluyendo acciones adicionales Lo anterior se da ya que Android solo puede poseer 4 procesos activos a la vez, pero posee una politica sumamente ágil para simular más de 4 procesos activos Cada vez que se genera un proceso nuevo y este pasa al tope de la pila, se guarda un

estado del proceso anterior y se ejecuta el nuevo Android guarda un estado de los procesos sumamente ligero solo contiene información de lo que el usuario estaba haciendo en ese momento Cuando ya existen 4 procesos y se abre un 5 automáticamente busca el menos utilizado y lo destruye En caso de que el usuario pulse el botón *Back* hasta llegar al proceso destruido Android toma el cache guardado reconstruye el proceso y lo muestra como si nada hubiese pasado De esta manera se puede simular una cantidad inmensa de procesos activos y mantener un óptimo rendimiento

Las aplicaciones Android pueden estar conformadas por los siguientes elementos

Activities Es la capa de presentación de la aplicación poseen una vista asociada, la cual poseerá la interfaz de usuario

Services Capa de background de nuestra aplicación realiza tareas de actualización notificación entre otras

Content Providers Permiten el intercambio de datos entre aplicaciones

Intents. Nos brinda el envío de mensajes entre las aplicaciones de la plataforma Generalmente utilizado para solicitar servicios que prestan otras aplicaciones

Broadcast Receivers Elemento que nos permite escuchar mensajes Intents provenientes de otras aplicaciones y realizar acciones personalizadas

Widgets Opción para integrar nuestras aplicaciones con el Home Screen de la plataforma cambiando contenido dinámicamente

Notificactions Con las mismas realizamos notificaciones al usuario sin interrumpir sus actividades actuales a través de ellas podemos notificar el estado o la respuesta de un servicio o un Broadcast Receiver

El ciclo de vida de una aplicación Android está determinado por los siguientes métodos

onCreate En este método se debe crear la interfaz que tendrá la actividad Es el evento indicado para inflar elementos de interfaz interactuar con Widgets y utilizado para acceder a los elementos del *layout* (apariencia de la aplicación)

onStart Se ejecuta cuando la actividad se está volviendo visible Se utiliza para inicializar variables de interfaz

onResume Es el lugar indicado para inicializar animaciones e inicializar el acceso a recursos de hardware (cámara GPS entre otros)

onPause Este método es llamado cuando la actividad está pasando a background es el momento perfecto para guardar datos persistentes antes de que la actividad pase a background

onStop Se habilita cuando la actividad ya no es visible para el usuario es probable que éste método nunca se ejecute ya que en caso de que el sistema posea baja memoria no podrá mantener la actividad en background y la destruirá automáticamente

onRestart Actua cuando la actividad es visible para el usuario nuevamente

onDestroy Ultimo método que ejecutará la actividad es el momento para realizar liberación y limpieza de recursos Se ejecuta cuando el sistema temporalmente destruye la instancia de la actividad para salvar espacio o si se ejecutó el método *finish* para distinguir cuál de los dos fue el causante se utiliza el método *isFinishing* Puede que este método nunca se ejecute ya que el sistema puede obviarlo en caso de memoria baja

A continuación en la (Fig 19) se muestra el diagrama de flujo del ciclo de vida de una aplicación Android

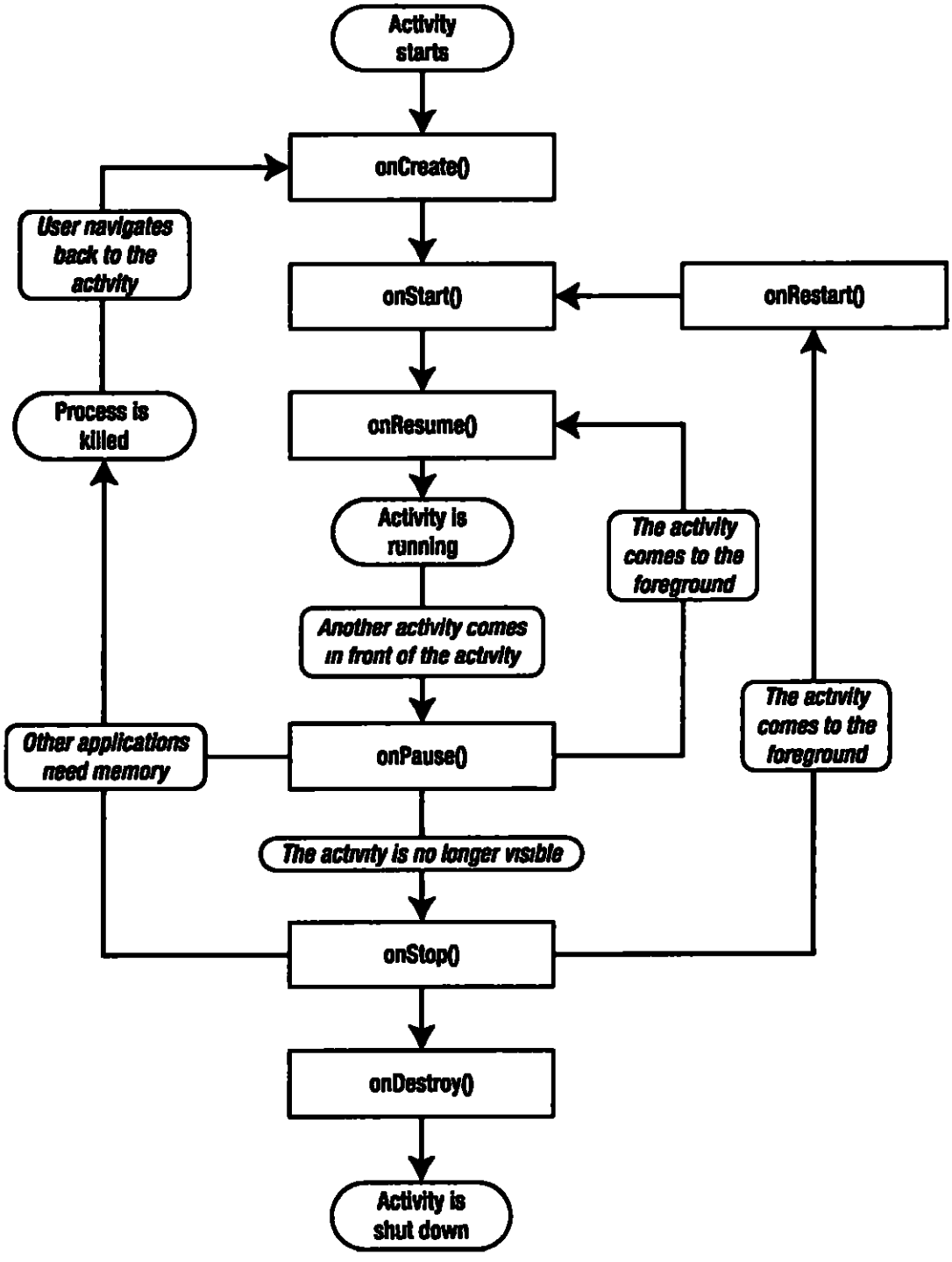


Fig 19 Ciclo de vida de una aplicación en Android

Una actividad puede encontrarse en uno de cinco estados posibles Sin embargo solo tres de estos estados son estáticos es decir la actividad puede permanecer en ellos por un periodo de tiempo prolongado A continuación se explican los estados

Active Cuando el Activity está en el tope de la pila posee todo el foco y captura los eventos del usuario debe mantenerse viva a toda costa Si la actividad pasa a segundo plano el sistema llama al método onPause()

Paused Hay veces que el Activity está visible pero no posee el foco del usuario Ocurre cuando se levanta un Activity que no cubre por completo el Activity que está debajo La actividad pausada no ejecuta ningun código y no puede interactuar con el usuario Android puede destruir un Activity en Paused para proveer recursos al Active Activity Si vuelve a primer plano se llama al método onResume() y se vuelve al estado de ejecución Si por el contrario la actividad pasa por completo a segundo plano el sistema llama a onStop() y transita al siguiente estado

Stopped La aplicación al caer en este estado queda completamente oculta en segundo plano y no es visible por el usuario Se conserva su instancia y toda la información correspondiente pero no puede ejecutar código El sistema llama al método onRestar() si la actividad vuelve a primer plano recuperando la información guardada Si la actividad es cerrada por completo se ejecuta el método onDestroy()

Inactive Cuando un Activity ha sido cerrado destruido o no ha sido lanzado aun Estos Activities no están en la pila

2 6 3 Archivos Importantes en un Proyecto Android

Al desarrollar un proyecto refiriéndonos al entorno Eclipse antes comentado el mismo crea diferentes archivos y carpetas que son importantes para el buen funcionamiento de la aplicación que deseamos desarrollar Muchos de estos archivos son creados con lineas de código pre cargado facilitando asi el desarrollo de la aplicación y estableciendo los parámetros básicos para su buen funcionamiento Estas líneas de código vienen escritas en lenguaje XML (lenguaje de marcas extensible) y Java Lo creado en XML es en general toda la apariencia que observara el usuario final de la aplicación y también es escrito en este lenguaje los permisos que dependiendo del alcance del desarrollo podamos necesitar como por ejemplo GPS (Sistema de posicionamiento global) Bluetooth Wi Fi (Fidelidad inalámbrica) y cualquier otro servicio nativo del sistema operativo o desarrollado por terceros Lo pre-cargado en código Java son las variables de la aplicación y los métodos o funciones que definirán el comportamiento de la aplicación Además todo lo que deseamos escribir como por ejemplo bucles animaciones variables nuevas rutinas nuevas clases en fin todo lo que requiere una decisión acción y reacción por parte de nuestra aplicación es escrito en lenguaje Java A continuación se mencionan los archivos más importantes a tener en cuenta a la hora de desarrollar una aplicación Android

2 6 3 1 Main xml

Android crea automáticamente un main xml en el directorio res/layout del proyecto Este archivo contiene código pre-cargado que define la estructura visual de la interfaz

usuario de la aplicación El entorno Eclipse facilita la edición de este archivo para agregar lo necesario a nuestra interfaz como por ejemplo botones campos de texto etc Si lo que se desea es crear interfaces animadas se recomienda entonces utilizar código Java

2 6 3 2 AndroidManifest xml

Cada proyecto debe tener un archivo llamado AndroidManifest xml en su directorio raiz Se trata de un archivo en lenguaje de código XML que mediante el uso de un sistema de etiquetas y atributos describe las caracteristicas fundamentales de la aplicación y define cada uno de sus componentes El sistema Android debe disponer de esta información antes de poder ejecutar cualquier código de la aplicación Algunas de las funciones que lleva a cabo este archivo son

Dar nombre al paquete Java Este nombre identifica la aplicación

Describe los distintos componentes de la aplicación actividades servicios clases etc

Determina los permisos que autoriza la aplicación para acceder a los distintos recursos del dispositivo

Establece la versión minima del sistema operativo Android que requiere la aplicación para funcionar correctamente

2 6.3.3 Nombredeaplicacion java

Se encuentra dentro del directorio *src* del proyecto y es creado con codigo pre cargado que contiene métodos que permiten lanzar la aplicación para mostrarla al

usuario En este archivo se escribe todo lo deseado que defina el comportamiento que queremos en la aplicación

A continuación en la (Fig 20) se muestra la estructura general de un proyecto en Android con la ayuda del entorno Eclipse

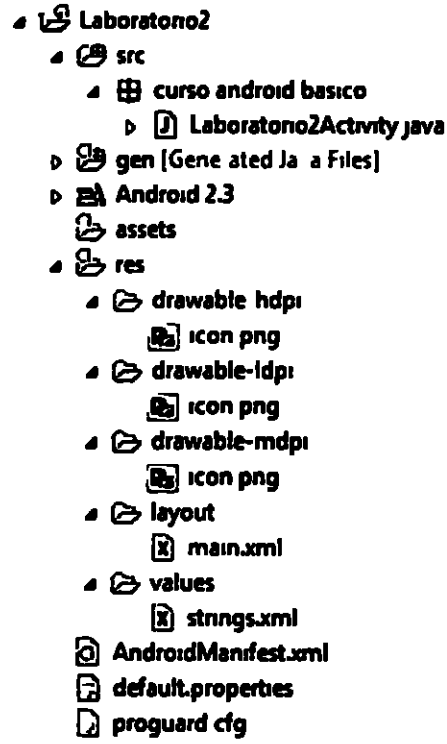


Fig 20 Estructura de un proyecto Android

3 ASPECTOS METODOLÓGICOS

Para desarrollar el prototipo del sistema, dividimos el desarrollo en tres partes fundamentales las cuales denominamos nodos Arduinos Apps Android e integración de las partes Lo que logramos dividiendo el desarrollo en partes es que cada una de ellas trabajará bien por si sola, para luego lograr acoplarlas entre si A continuación describimos cada una de las partes mencionadas anteriormente

3 1 Nodos Arduinos

Para desarrollar esta etapa de la investigación tuvimos que seccionarla en tres partes a las que denominamos nodo principal nodos secundarios e integración de los nodos A continuación explicaremos las tres partes antes mencionadas

3 1 1 Nodo Principal

Los objetivos del nodo principal son los siguientes

Controlar el sistema de riego de forma autónoma

Mantenerse comunicado con los nodos secundarios para la obtención de datos

Comunicarse con la App Android llamada AgroServer para enviar los datos capturados y recibir datos de control

Para lograr los objetivos del nodo principal se investigó cuáles eran los elementos que nos permitirían llegar a esto Dichos elementos cumplen los siguientes aspectos bajo

costo alta disponibilidad en el mercado utilidad experimental buen desempeño de funcionamiento A continuación describimos las caracteristicas técnicas y las configuraciones de los elementos que conforman el nodo principal

3 1 1 1 XBee Serie 2

Al principio de la investigación se pensó desarrollar un prototipo con sensores de humedad de suelo conectados por medio de alambres con ello surgieron dudas y posibles fallos e incomodidades futuras Por ello se decidió buscar la viabilidad de una tecnologia inalámbrica como lo es XBee de esta forma evitamos posibles fallos producidos como por ejemplo roedores desconexiones por circulación de personas en el sembradio etc

Las caracteristicas técnicas las podemos ver en el (Cuadro 2 pag 26) Decidimos elegir este tipo de tecnologia inalámbrica, debido a su bajo consumo energético comparado con otras tecnologias como por ejemplo Wi Fi y Bluethoot Otro punto a destacar es su escalabilidad ya que la serie 2 de estos módems puede alcanzar distancias en la actualidad de hasta 28 millas (https //www sparkfun com/pages/xbee_guide) de coordinador a terminal logrando ampliar nuestra red para cubrir mayores sembradios

El modem XBee Serie 2 se encargá de mantener comunicado nuestro nodo principal con los demás nodos secundarios a través de él podemos enviar y recibir datos para luego tomar una decisión si encender o apagar el riego

El firmware del modem en discusión es de tipo coordinador y en modo API cargado con el software X CTU ya que será el encargado de organizar nuestra red XBee El modo API nos permite realizar la comunicación del coordinador con todos los XBee de la red es decir una red multipuntos Por ultimo es necesario configurar a través del X CTU

el PAN ID que será el numero que describirá nuestra red XBee A continuación en la (Fig 21) se muestra la captura del firmware cargado al modem XBee conectado a nuestro nodo principal

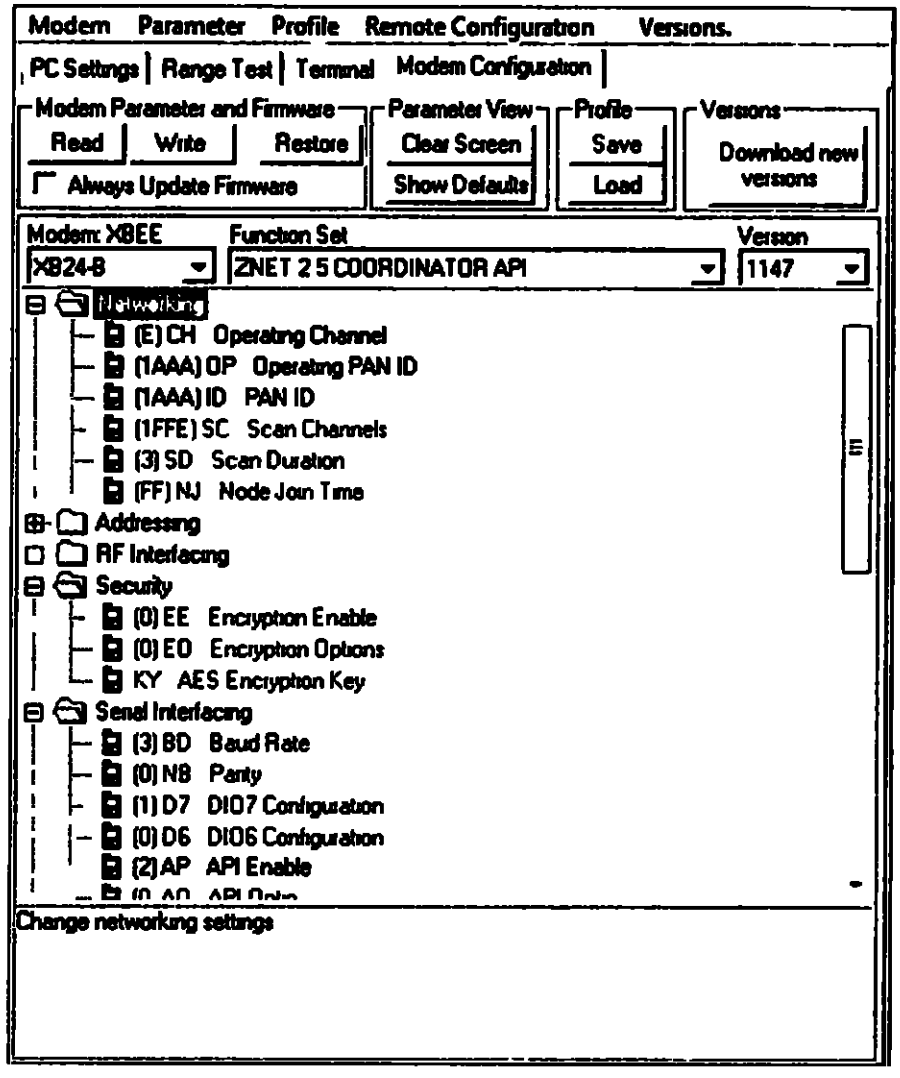


Fig 21 Firmware coordinador

La libreria utilizada para trabajar con nuestros módulos XBee en el entorno Arduino fue creada por Andrew Rapp y es de acceso libre llamada *xbee arduino* versión 0 3 Se puede encontrar en la siguiente página (code google com/p/xbee arduino)

3 1 1 2 Wireless SD Shield

Utilizado para mayor comodidad de conexión entre el Arduino y el módulo XBee También es necesario para configurar nuestros módulos XBee ya que contiene un interruptor que consta de dos modos micro y USB Para configurar el módulo XBee a través del software X CTU es necesario colocar este interruptor en modo USB Sobre este shield es montado el modem XBee para luego montar el shield en el Arduino y de esta forma lograr la conexión entre ellos Su consumo energético es de 3 3 voltios de corriente directa A continuación en la (Fig 22) se muestra el aspecto del shield comentado

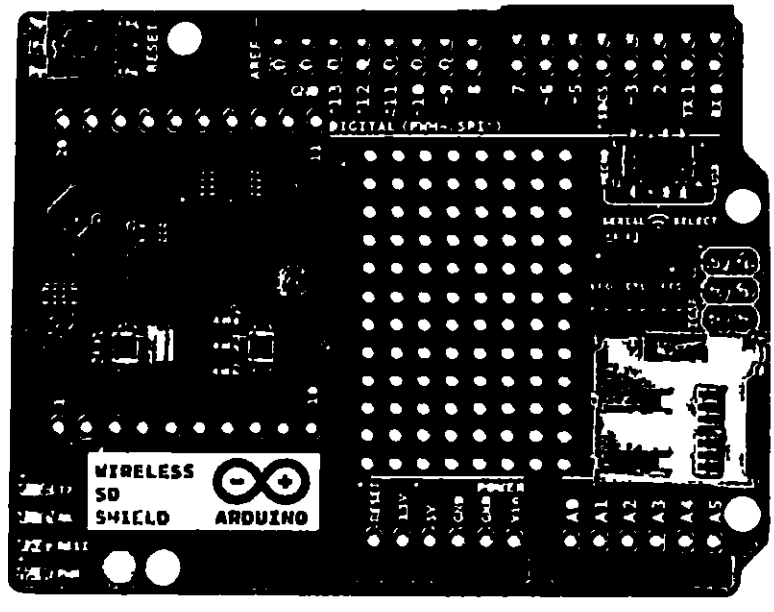


Fig 22 Wireless SD Shield

3 1 1 3 Tarjeta de Relay

Manipulará el encendido y apagado del sistema cuando el micro-controlador de nuestro nodo principal lo indique Esta tarjeta consta de entradas o disparadores que requieren una excitación de 5 voltios de corriente directa para activar el relay que realizara el control El relay será capaz de soportar un paso de 250 voltios de corriente alterna y 10 amperios suficiente para controlar una válvula de tipo solenoide o bien una bomba de riego eléctrica A continuación en la (Fig 23) se muestra el aspecto de la tarjeta de relay utilizada en esta investigación

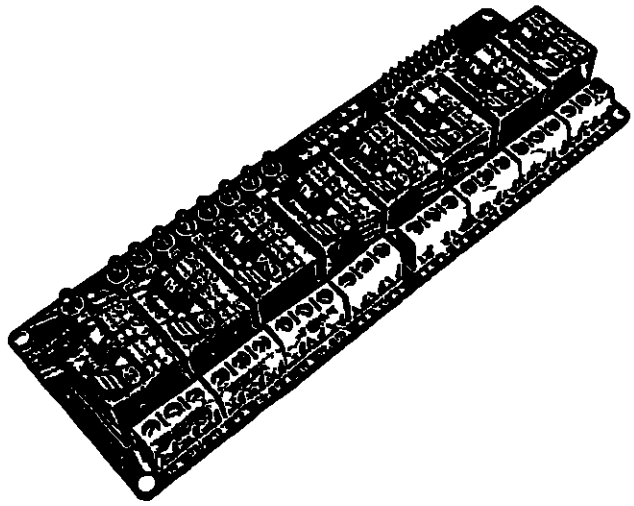


Fig 23 Tarjeta de relays

La tarjeta de relay utilizada en esta investigación consta de 8 relays cada uno con las mismas especificaciones descritas anteriormente

3 1 1 4 Sensor de Flujo

Su función será la de monitorear el flujo de agua por la tubería principal del sistema de riego luego de que se haya encendido el sistema Cuando el micro-controlador da la orden para arrancar el riego a través de la tarjeta de relay el mismo procede a verificar la lectura del sensor de flujo si la lectura es de 4 5 voltios esto quiere decir que el sistema tiene un funcionamiento normal pero si la lectura es 0 voltios entonces existe un problema en el sistema y el micro-controlador tomara la decisión de apagar el sistema El sensor de flujo trabaja con un rango de alimentación de 5 a 18 voltios corriente continua y 15mA de corriente máxima A continuación en la (Fig 24) se muestra la apariencia del sensor de flujo

Fig 24 Sensor de flujo

3 1 1 5 Buzzer

Será nuestra bocina que indicará la alarma del sistema de forma audible Cuando se produzca el problema de falta de flujo de agua por la tubería principal del sistema el micro-controlador aparte de apagar el riego activará la bocina indicando el problema antes mencionado Su alimentación energética es en el rango de 3 a 5 voltios corriente directa y su sonido es controlado por modulación de ancho de pulso (PWM)

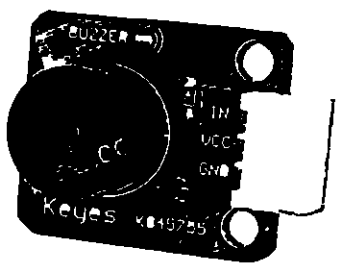


Fig 25 Buzzer

3 1 1 6 Liquid Crystal Display 16x2

LCD 16x2 (LCD 16x2 (16 caracteres horizontales por 2 filas verticales) indica los siguientes parámetros del sistema porcentaje de humedad del suelo estado del sistema (ON/OFF) estado de la alarma (ON/OFF) y modo del sistema (automático/manual) Su consumo energético es de 5 voltios corriente directa y un máximo de 1 amperio A continuación en la (Fig 26) se muestra el aspecto de la pantalla comentada

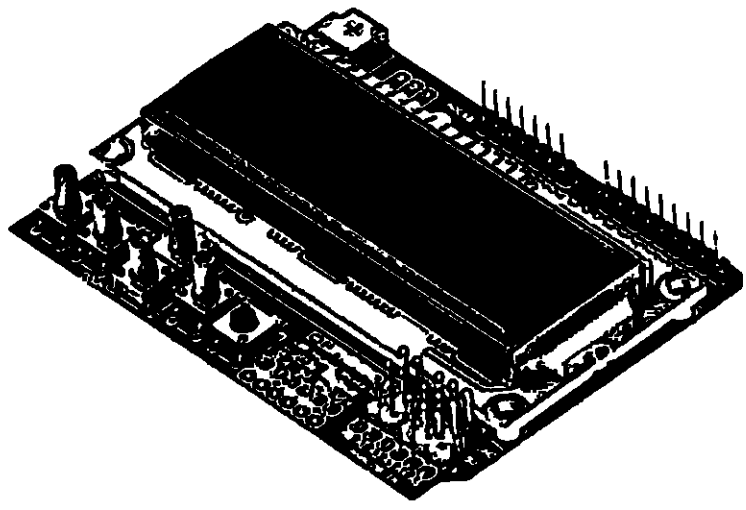


Fig 26 LCD 16x2

3 1 1 7 Arduino Mega ADK

Es el micro-controlador de nuestro nodo principal se comunica con todos los nodos secundarios para solicitar los datos de humedad y poder de esta forma tomar la decisión de arrancar o apagar el riego También mantiene una comunicación con un dispositivo Android si esta conectado a él a través de su puerto USB tipo A Gracias a este micro controlador logramos la automatización del sistema Las características técnicas se explican en la sección fundamentación teórica

Al principio del desarrollo se pensó utilizar un dispositivo PLC (Controlador logico programable) para realizar la automatización del sistema de riego Estos dispositivos manejan altos niveles de corriente y voltajes para controlar cualquier tipo de elemento que funcione con corriente alterna muchos modelos de ellos sin la necesidad de adaptar otro dispositivo externo A medida que investigábamos sobre ellos descubrimos algunos tipos de limitantes en cuanto a darle una orientación a nivel de investigación y desarrollo

destacando también su alto costo en el mercado Los PLC en su gran mayoria tienen entornos de programación propietarios y lenguajes de programación poco convencionales limitando de esta forma a los investigadores y desarrolladores que desean integran multiples plataformas al igual que lenguajes de programación Para lograr ampliar las prestaciones de los PLC es necesario comprar un dispositivo de la misma marca o encontrar algun tipo de compatibilidad con marcas estándar lo cual es muy complicado Un PLC básico de una marca prestigiosa como SIEMENS especificamente el modelo SIMATIC S7 200 cuesta alrededor de B/ 400 00 dólares y si además deseamos una comunicación inalámbrica con el mismo necesitamos adquirir otro dispositivo con costo aproximado de B/ 100 00 especificamente el modem modelo TC65T pero si pensamos en multiples nodos comunicándose inalámbricamente el costo no sería viable para ningun agricultor promedio Por estas razones decidimos descartar el uso de un PLC para el desarrollo de esta investigación (Schugurensky y Capraro (2008))

Arduino en la actualidad es una plataforma próspera gracias a su orientación de software y hardware libre con muchas personas trabajando alrededor del mundo con él Cada dia se publican en internet nuevos videos papers dispositivos externos modelos Arduinos con mayor capacidad de procesamiento (por ejemplo el mas reciente Arduino Intel Galileo) librerías entornos de programación tutoriales prototipos de soluciones para empresas y miles de proyectos que trabajan con Arduino También tenemos que destacar que hoy día no existe algun indicio de que Arduino saldrá del mercado o pasará de moda por estas razones elegimos como controlador de nuestro prototipo a la plataforma Arduino

Arduino Mega ADK nos brinda una ventaja en comparación con los demás micro controladores de esta plataforma, comunicarnos directamente y fácilmente con dispositivos Android como se especificó en la sección fundamentación teórica Gracias a lo mencionado anteriormente los datos capturados por el ADK los podemos enviar a nuestra App Android AgroServer y recibir órdenes de control de ella Por esta utilidad elegimos el Arduino Mega ADK para nuestra investigación

El sketch desarrollado para el Arduino Mega ADK lo podemos observar en el Anexo A Dicho sketch consta de variables que son necesarias cambiar dependiendo del numero de nodos que tendrá nuestra red y del tipo de cultivo En el desarrollo del prototipo se utilizaron dos nodos un nodo principal y un nodo secundario pero este prototipo es util para implementar multiples nodos solo basta con agregar las direcciones de los nuevos módems XBee en la sección del código comentada como *SH and SL addresses* luego se añaden estas direcciones al vector *sh* y *sl*' localizado en la misma sección del código antes mencionada con ello ampliamos nuestra red tanto como la tecnología XBee nos lo permita siendo esto comentado en la sección fundamentación teórica Otra variable a cambiar en el sketch es la llamada *humedadreferencia* ella se encarga como su nombre lo indica de la referencia de humedad que se debe mantener en el suelo para un determinado cultivo por parte de nuestro prototipo para ello previamente nos debemos apoyar en un ingeniero agrónomo o algun agricultor experimentado que sepa la humedad promedio que debe ser brindada al cultivo En nuestra investigación decidimos utilizar una humedad de referencia de un 50% ya que en general se sugiere un valor de criterio de riego (CR) del cincuenta por ciento (CR = 0 5) para la mayoria de cultivos pero si el cultivo sufre de déficit de agua entonces se le asigna valores de CR del sesenta porciento

(CR = 0 6) y para cultivos que soportan de mejor manera un estrés hídrico se le asigna un CR = 0 3 (Ramírez y Valenzuela (1998))

A continuación en la (Fig 27) se muestran los criterios de riego de una gran cantidad de cultivos

Cultivo	CR
Ají	0 5
Ajo	0 5
Alcachofa	0 50
Alfalfa	0 65
Arveja	0 6
Brocoli	0 5
Cebolla	0 50
Coliflor	0 5
Damascos	0 65
Duraznos	0 65
Espárragos	0 5
Empastadas	0 65
Frejol	0 50
Frutilla	0 5
Habas	0 6
Lechugas	0 40
Maíz	0 65
Manzanos	0 65
Melón	0 5
Menta	0 35
Papas	0 30
Perales y ciruelos	0 65
Pimiento	0 5
Repollo	0 5
Sandía	0 5
Trigo invierno	0 65
Trigo primavera	0 65
Vid	0 65
Zanahoria	0 5
Zapallo	0 5

Fig 27 Criterio de riego de los cultivos (Ramírez y Valenzuela (1998))

Los datos capturados por el Arduino Mega ADK son porcentaje de humedad del suelo estado del sistema (ON/OFF) estado de la alarma (ON/OFF) y modo del sistema (automático/manual) Los mismos son almacenados en un vector llamado *escritura* para luego ser impreso en la pantalla LCD 16x2 y enviados a la App AgroServer si se encuentra conectado un dispositivo Android al ADK de esta forma el agricultor podrá observar el funcionamiento del sistema

Los porcentajes de humedad del suelo son capturados a través de nuestros nodos secundarios y enviados por medio de los módems XBee El nodo principal se encargará de enviar peticiones a cada nodo secundario solicitando la humedad del suelo de dicha área del cultivo este a su vez al enviar la solicitud queda a la espera de la confirmación de recepción por parte del nodo secundario el nodo secundario al recibir la petición envía al nodo principal la confirmación de recepción de la solicitud el nodo secundario procede a leer el sensor de humedad de suelo (Soil Moisture Sensor) prepara el paquete y lo envía al nodo principal este a su vez queda a la espera de la recepción del paquete por parte del nodo principal cuando el nodo principal recibe el paquete envia la confirmación al nodo secundario y comienza nuevamente el proceso con otro nodo secundario Este proceso lo realiza en aproximadamente 100ms Luego de capturar todos los datos de humedad de suelo de los nodos secundarios se promedia el porcentaje de humedad total y se escribe en el vector *escritura* de nuestro sketch

A continuación en la (Fig 28) se muestra el diagrama de flujo del funcionamiento del sketch cargado en el Arduino Mega ADK que utilizamos para nuestro desarrollo

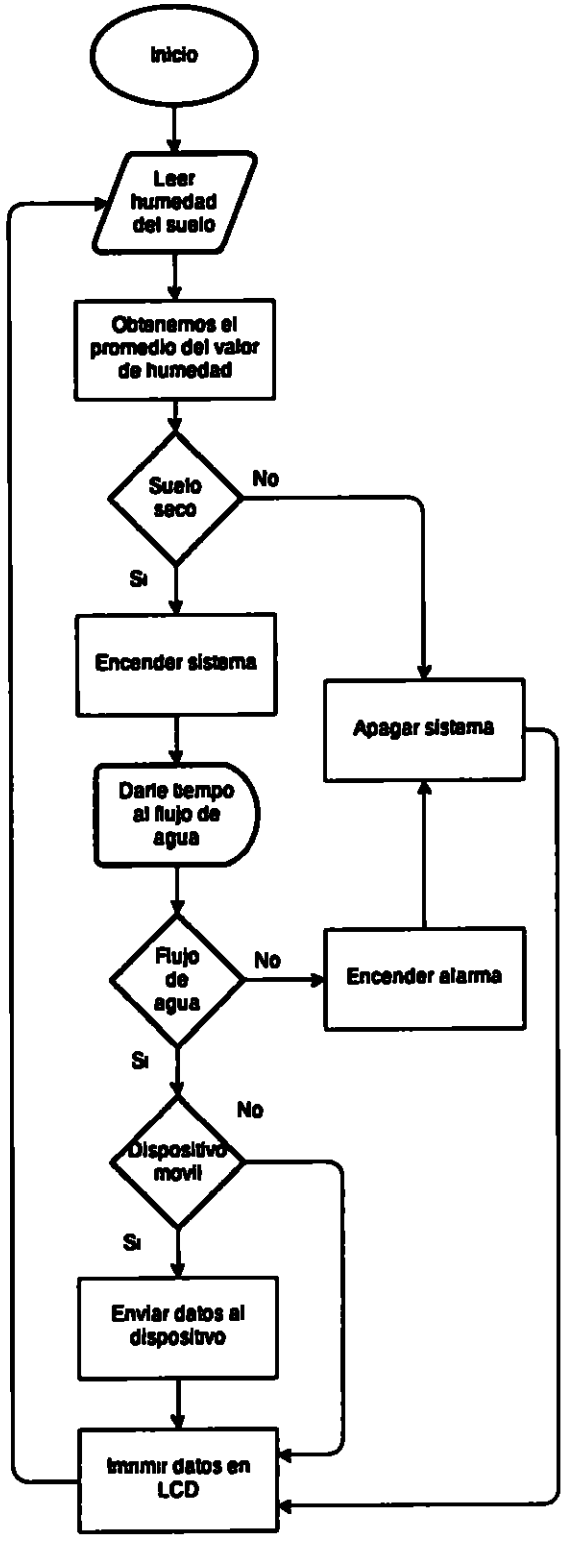


Fig 28 Diagrama de flujo del sketch Arduino Mega ADK

3 1 2 Nodos Secundarios

En la investigación se utilizó un solo nodo secundario debido a que no estaba contemplado desde los inicios utilizar XBee como tecnología de comunicación por ello se buscó apoyo en el centro de innovación tecnológico de informática y comunicaciones de la Universidad de Panamá (CITIC UP) que nos facilitó los módems XBee que tenían disponibles para desarrollar la investigación

Los objetivos del nodo o nodos secundarios son los siguientes

Leer la humedad del suelo a través del soil moisture sensor que explicaremos más adelante

Recibir peticiones del nodo principal por medio del modem XBee

Enviar el porcentaje de humedad del suelo al nodo principal mediante XBee

El nodo secundario utilizado en esta investigación está formado por algunos elementos presentes en el nodo principal los cuales son Wireless SD Shield y XBee serie 2 Los demás elementos presentes en el nodo secundario procedemos a detallarlos a continuación

3 1 2 1 Soil Moisture Sensor

Es un tipo de sensor resistivo que utiliza dos sondas para medir la corriente que circula en el medio a través de la cual podemos determinar la resistencia con lo cual se calcula la humedad del terreno El porcentaje de humedad será mayor mientras más cantidad de agua esté presente en la tierra Los valores capturados son de tipo análogo enviando un valor entero de 0 a 950 Si el suelo está seco obtendremos un valor entero de

0 a 300 si esta humedo de 301 a 600 y si está saturado de 601 a 950 Su consumo energético es de 3 3 a 5 voltios de corriente directa y 35 milis amperios máximo En la (Fig 29) se muestra la apariencia del sensor

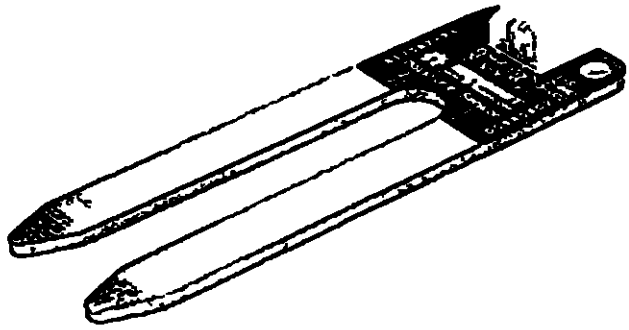


Fig 29 Soil moisture sensor

3 1 2.2 XBee Serie 2

De este modem ya hemos comentado anteriormente pero el mismo en el nodo secundario se le debe cargar otro firmware que sea de tipo *Router/End Device* por medio del software X CTU también en modo API Aparte de cargarle el firmware correspondiente es necesario también editar el numero PAN ID que corresponde a la red y que fue introducido al modem coordinador este numero debe ser el mismo tanto en el coordinador como en el nodo o nodos secundarios A continuación en la (Fig 30) se muestra la captura del firmware cargado al módem XBee secundario

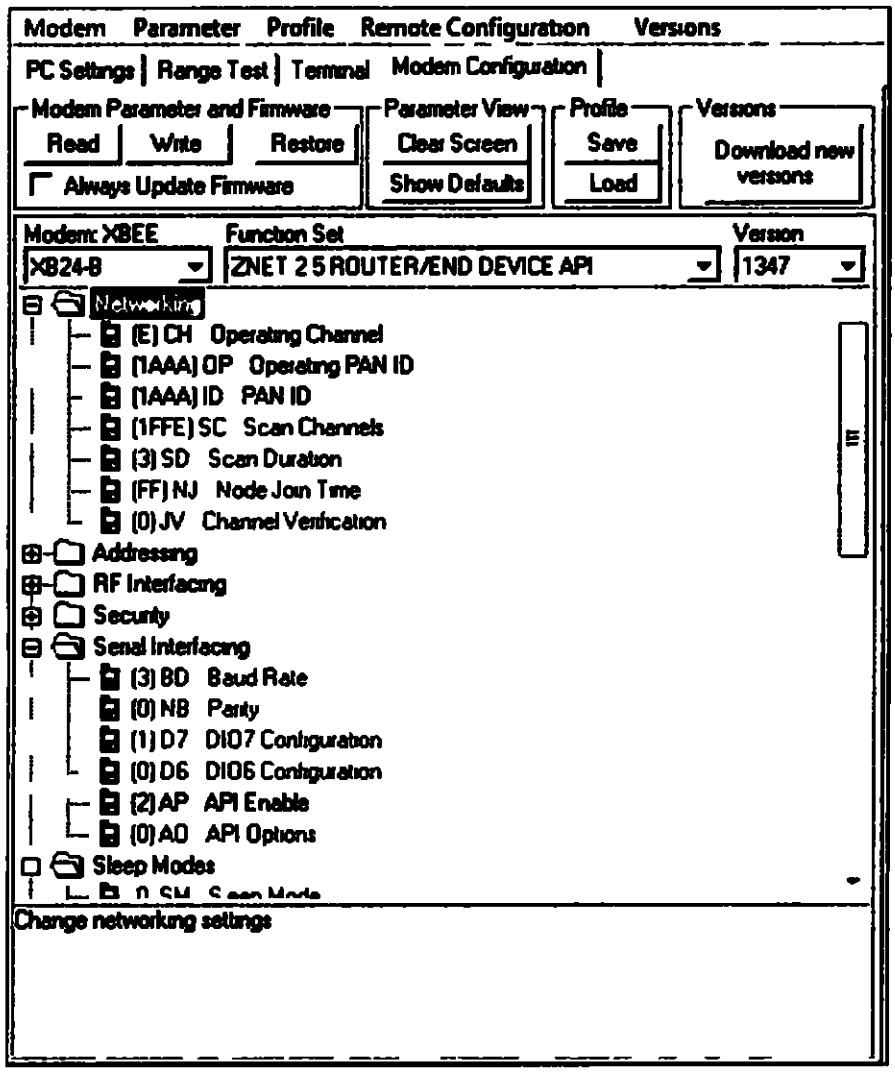


Fig 30 Firmware Router/End Device

3 1 2.3 Arduino Uno

Uno de los modelos Arduinos más utilizados y el adecuado para comenzar con todo lo relacionado a la plataforma Arduino A recibido varias actualizaciones y la más reciente es la revisión numero tres (R3) que es la que utilizamos en esta investigación Decidimos elegir este modelo como nodo secundario de nuestro sistema, ya que no es necesario grandes capacidades de memoria y procesamiento para ejecutar las instrucciones de este

nodo Tendrá conectados los siguientes elementos modem XBee Wireless SD Shield y el soil moisture sensor Se encargará de capturar el valor de humedad a través del sensor y enviará este dato por medio del módulo XBee al nodo principal

El sketch cargado al nodo secundario se puede observar en el Anexo B Funciona de la siguiente forma el nodo secundario espera la solicitud del nodo principal al llegar envia una confirmación de la recepción de la solicitud luego lee el soil moisture sensor para luego empaquetar el dato y enviarlo al nodo principal a su vez queda a la espera de la confirmación de la llegada del paquete al nodo principal Debido a que el soil moisture sensor arroja valores entre 0 y 950 decidimos realizar una regla de tres para representar el valor de humedad en porcentaje A continuación muestro la operación en la ecuación (7)

(7) $$\%humedad = humedad\ capturada * \frac{100}{950}$$

Otro aspecto a destacar en el sketch es que cada nodo secundario tiene que saber la dirección sh y sl del coordinador por lo tanto cuando se cambie el coordinador en el sistema también se tiene que actualizar la variable sh y sl en el sketch de todos los nodo secundarios A continuación en la (Fig 31) se muestra el diagrama de flujo del sketch cargado en el nodo secundario

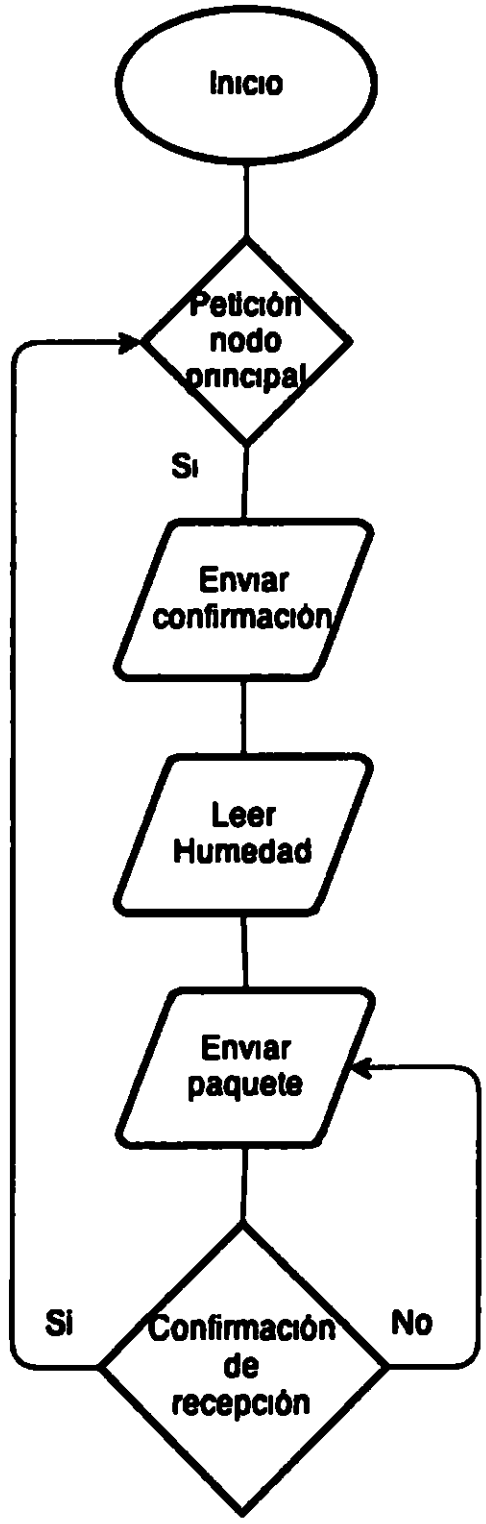


Fig 31 Diagrama de flujo del sketch del nodo secundario

3 1 3 Integración de los Nodos

Cuando logramos el funcionamiento estable de cada nodo procedimos a realizar la integración de esta forma nos aseguramos que cada nodo trabaja de forma estable por si solo y que si surgia algun problema en la integración seria error de comunicacion XBee y no por motivos de los micro-controladores de esta forma nos enfocaríamos en todo caso en solucionar la comunicación XBee Gracias a la estabilidad de los módems XBee y la librería de acceso libre llamada *xbee arduino* comentada en puntos anteriores no hubo mayor problema en lograr el intercambio de datos entre los nodos Si se cargan los firmwares correspondientes a cada modem y se manejan las direcciones sh y sl en cada sketch como fue comentado en los puntos anteriores no debe surgir ningun problema en el intercambio de datos entre los nodos Al final lo que se logra integrando los nodos es lo mostrado en el esquema de la (Fig 32)

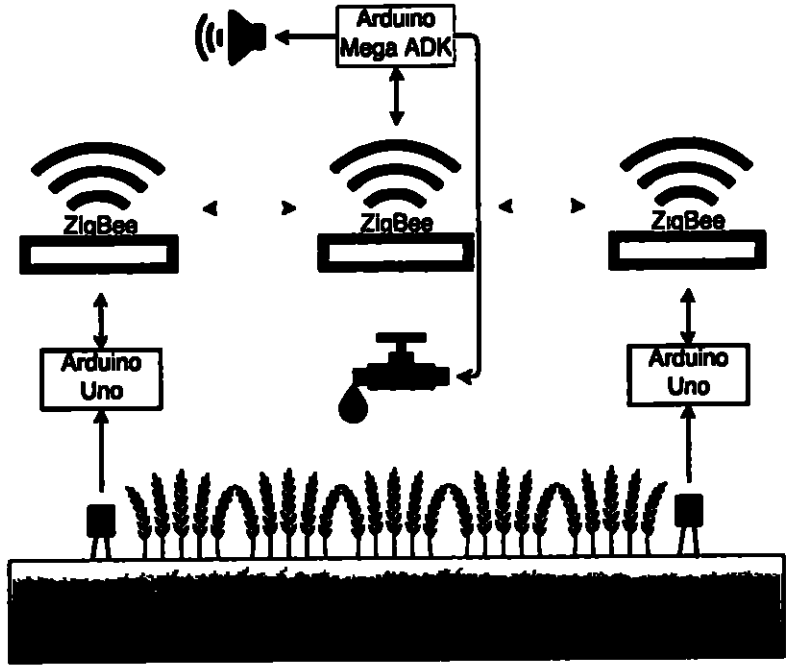


Fig 32 Esquema de la integración de nodos

3 2 Apps Android

Decidimos utilizar la plataforma Android principalmente por su desarrollo de código abierto debido a esta caracteristica existe a nivel mundial muchisimas personas trabajando en él creando aplicaciones y ofreciéndolas muchas de ellas de forma gratuita logrando así expandir las funcionalidades del sistema operativo y utilizar dichas aplicaciones para incrementar aun más el potencial de nuestra propia aplicación sin contar los diferentes servicios que Google presta de forma gratuita a los desarrolladores

Se desarrollaron dos aplicaciones Android en la investigación una llamada *AgroServer* y la otra con nombre *AgroUser*' A continuación se comentara el esquema de funcionamiento de cada una

3 2 1 AgroServer

Los objetivos que cumple esta aplicación son los siguientes

Recibir los datos del Arduino Mega ADK y mostrarlos al agricultor a través de un dispositivo móvil (Tablet o smartphone)

Escribir datos capturados en un archivo en la nube Google Drive llamado AgroServer

Enviar datos de control al Arduino Mega ADK

Recibir SMS de control de la aplicación *AgroUser*' y transmitirlos al Arduino Mega ADK siendo un puente entre ambas plataformas

El primer objetivo comentado anteriormente es el más importante de la aplicación si no se logra realizar una conexión con el Arduino Mega ADK la aplicación es

despreciable debido a que este enlace es el encargado del monitoreo y control remoto del nodo principal del sistema Para lograr el enlace entre ambas plataformas es necesario seguir una serie de pasos en la creación de la aplicación que se detallan en el libro llamado *Beginning Android ADK with Arduino* (Böhmer (2012)) y por ello no se entrara en detalle sobre el tema Cabe destacar que para que el enlace funcione es necesario realizar cambios tanto en el sketch Arduino como en la aplicación Android y hacer coincidir ciertas líneas de código por lo tanto este proceso se puede hacer tedioso si no se realiza con orden

En la aplicación existen dos objetos de tipo *BroadcastReceiver* uno de ellos nos brinda la caracteristica de recibir cualquier entrada del conector USB del dispositivo Android y otro está pendiente de cualquier mensaje de texto (SMS)

BroadcastReceiver USB Si se conecta un dispositivo USB al Android y este converge con las lineas de código compatibles con la aplicación *AgroServer* el sistema Android muestra la opción al usuario en pantalla para abrir dicha aplicación esto quiere decir que ambas plataformas se reconocieron entre si y se puede establecer el enlace En nuestro caso los que nos interesa es que el Arduino Mega ADK sea reconocido por nuestra aplicacion

BroadcastReceiver SMS Cuando el sistema operativo Android recibe un mensaje SMS emite un mensaje publico que cualquiera aplicación instalada en el dispositivo es capaz de interpretar si contiene un objeto de tipo *Broadcast* especificamente para recibir SMS De esta forma nuestra aplicación tiene la capacidad de extraer de cualquier SMS el numero del emisor y el mensaje

Gracias a estos dos objetos Broadcast obtenemos el enlace con el nodo principal y el control remoto del sistema por medio de SMS A continuación en la (Fig 33) se muestra el diagrama de flujo al recibir un SMS

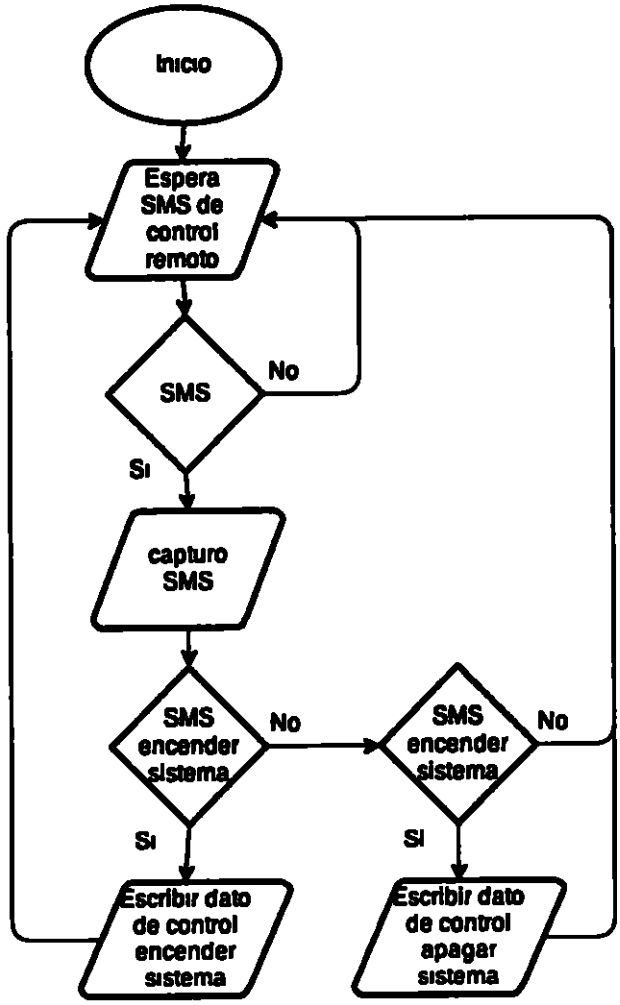


Fig 33 Recepción de SMS

Al abrir *AgroServer'* se va realizando una serie de procesos ocultos al usuario El primer proceso es establecer el enlace USB entre el Arduino Mega ADK y el dispositivo Android Luego la aplicación queda a la espera de cualquier SMS en cualquier momento de la ejecución si este contiene el numero telefónico correspondiente del emisor

AgroUser' y un valor de 1 o 0 entonces se procede a enviar dicho valor como dato de control al Arduino ADK para encender o apagar el sistema respectivamente Continuando con la ejecución de la aplicación la misma verifica si existe alguna cuenta Google almacenada en las preferencias de la aplicación si no existe ninguna cuenta vinculada a la aplicación entonces se muestra en pantalla el *Manager Account'* del sistema operativo Android mostrando todas las cuentas almacenadas en el Android y el usuario solo tendrá que elegir una cuenta Google a su vez esta cuenta se guardara en las preferencias de la aplicación y cuando se vuelva a ejecutar la aplicación en otro momento ya no será necesario pasar por el proceso antes comentado simplemente la aplicación extraerá la cuenta de las preferencias y la utilizara para poder acceder luego a Google Drive Cuando se vincula una nueva cuenta a la aplicación se le solicita al usuario ceder los permisos para que la aplicación pueda hacer uso del espacio en la nube Google Drive y manipular archivos *Spreadsheet* Una vez el usuario haya realizado lo anterior se procede a ingresar al espacio en la nube Google Drive del usuario en busca de un *Spreadsheet* con nombre *AgroServer'* si dicho archivo no existe en la cuenta del usuario el mismo se crea automáticamente Este va a ser el archivo donde se escribirán todos los datos del sistema

El objetivo de elegir Google Drive y un archivo Spreadsheet para almacenar los datos es debido a su flexibilidad y costo Cualquiera puede tener en la actualidad una cuenta Gmail que sirve para todos los servicios que brinda Google de forma gratuita incluyendo Google Drive Spreadsheet fue la elección debido a que es una hoja de cálculo gratuita con una capacidad para ingresar 400 000 celdas de datos lo suficiente para escribir todos los datos capturados durante el desarrollo de un cultivo (Gutiérrez et al (2009)) Además

en una hoja de cálculo podemos analizar directamente los datos de nuestro sistema con ayuda de gráficas y operaciones matemáticas Por ultimo y como aspecto innovador como el Spreadsheet estará en Google Drive tenemos acceso al archivo en cualquiera parte del mundo donde exista internet y podamos ingresar a nuestra cuenta, por lo tanto gracias a Google Drive y su Spreadsheet logramos el monitoreo remoto del sistema

Para poder trabajar con Google Drive y su aplicación Spreadsheet se tuvieron que realizar algunos cambios en el entorno de programación Eclipse además de seguir una serie de pasos y comprender una serie de ejemplos que se explican en la página oficial del SDK llamado *Google Drive SDK*" (https //developers google com/drive/web/) por ello no se entrará en mayores detalles

Una vez la aplicación junto con el usuario realizaron todos los procesos antes mencionados entonces es posible leer los datos que provienen del Arduino Mega ADK y mostrarlos en pantalla En la (Fig 34) se muestra la ventana principal de la aplicación

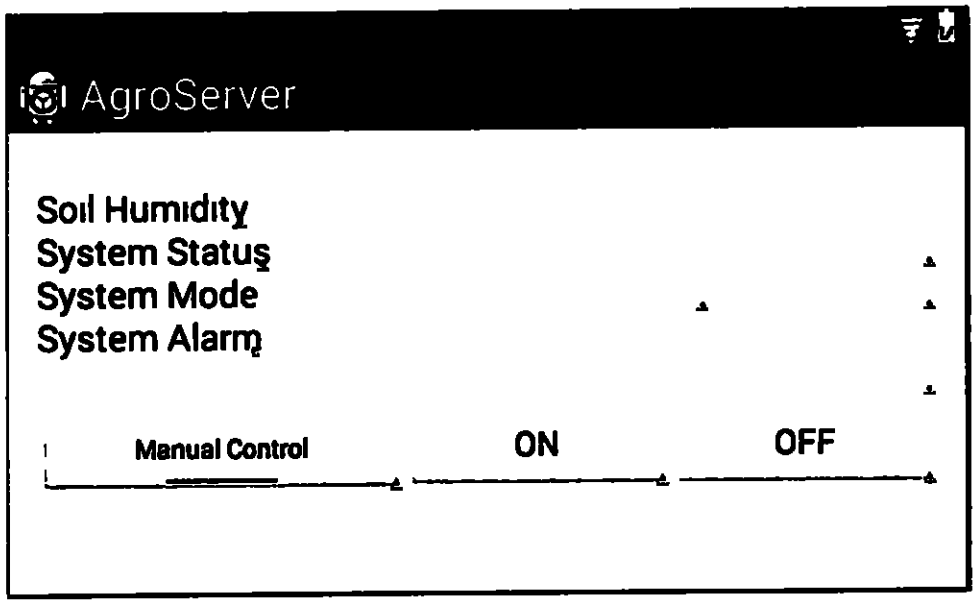


Fig 34 Aplicación AgroServer

Los datos enviados por el Arduino Mega ADK son porcentaje de humedad de suelo estado del sistema (ON OFF) Modo del sistema (Automatic Manual) alarma del sistema (ON OFF) Estos datos son mostrados en pantalla en su correspondiente renglón de la (Fig 34) Los botones con nombre Manual Control ON y OFF se utilizan para cambiar el sistema a modo manual Por defecto el sistema es autónomo pero si en algun momento el agricultor desea activar el sistema o apagarlo por alguna razón el mismo tendrá que presionar el botón Manual Control para habilitar los otros dos botones y poder entonces enviar la orden al Arduino Mega ADK de encender o apagar el sistema con el botón correspondiente Para salir del control manual solo basta presionar nuevamente el botón Manual Control y el sistema retomara el control automático A continuación en la (Fig 35) se muestra el diagrama de flujo de la pantalla principal de la aplicación AgroServer

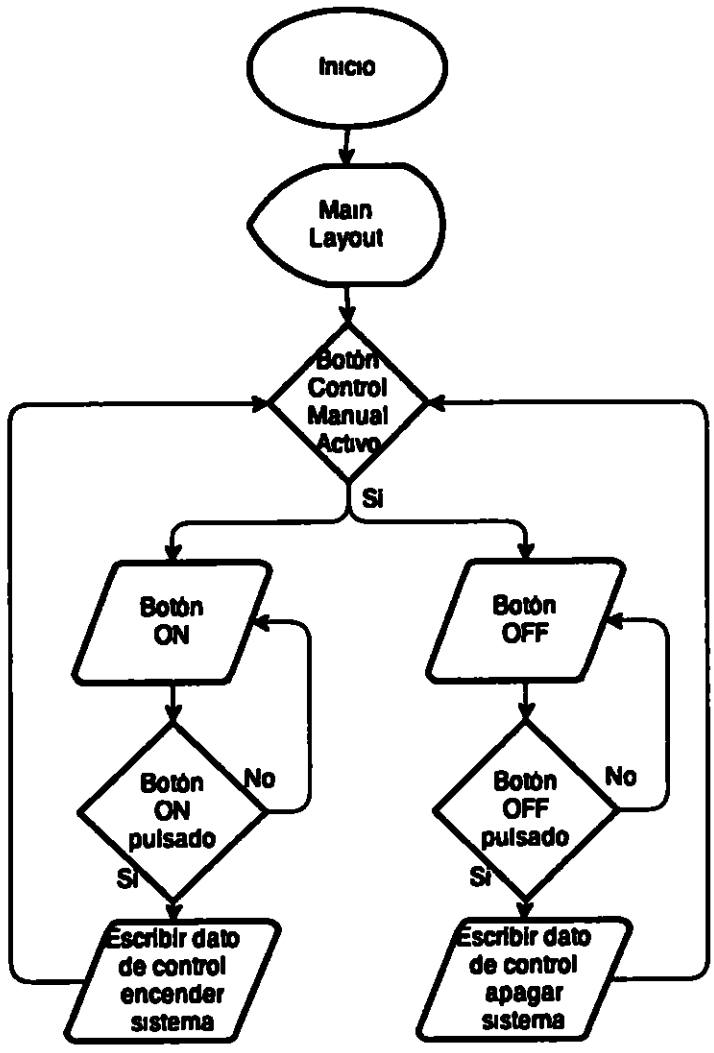


Fig 35 Main layout AgroServer

Cuando los datos son mostrados en pantalla se procede por medio de un proceso oculto al usuario o hilo escribir estos datos en el archivo almacenado en Google Drive llamado *AgroServer* El ciclo de la aplicacion consiste en leer los datos mostrar los datos en pantalla y escribir los datos en el archivo en la nube a su vez está pendiente de cualquier SMS de control entrante o acción del usuario con los botones

Todas las aplicaciones Android al crearlas tiene por defecto un menu de opciones que se ejecuta cuando se presiona el botón de ajustes del dispositivo Android Este menu fue utilizado para brindarle dos opciones al usuario una de ellas es lanzar en pantalla nuevamente el *Account Manager*' para vincular otra cuenta Google a la aplicación descartando la anterior y la otra opción es mostrar una pantalla con la página de inicio de Google Drive para de esta forma poder en la misma aplicación monitorear el sistema y observar los datos almacenados en el archivo A continuación en la (Fig 36) se muestra el diagrama de flujo del menu comentado

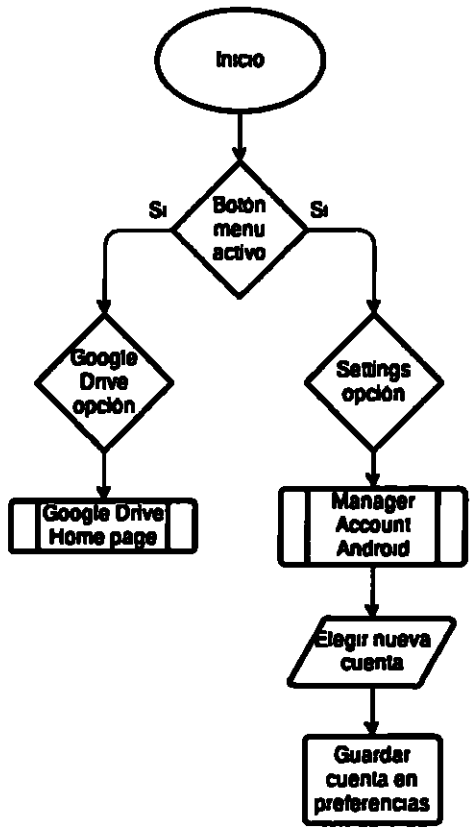


Fig 36 Diagrama de flujo Settings Menu

En la (Fig 37) se puede observar el diagrama de flujo general de la aplicación AgroServer

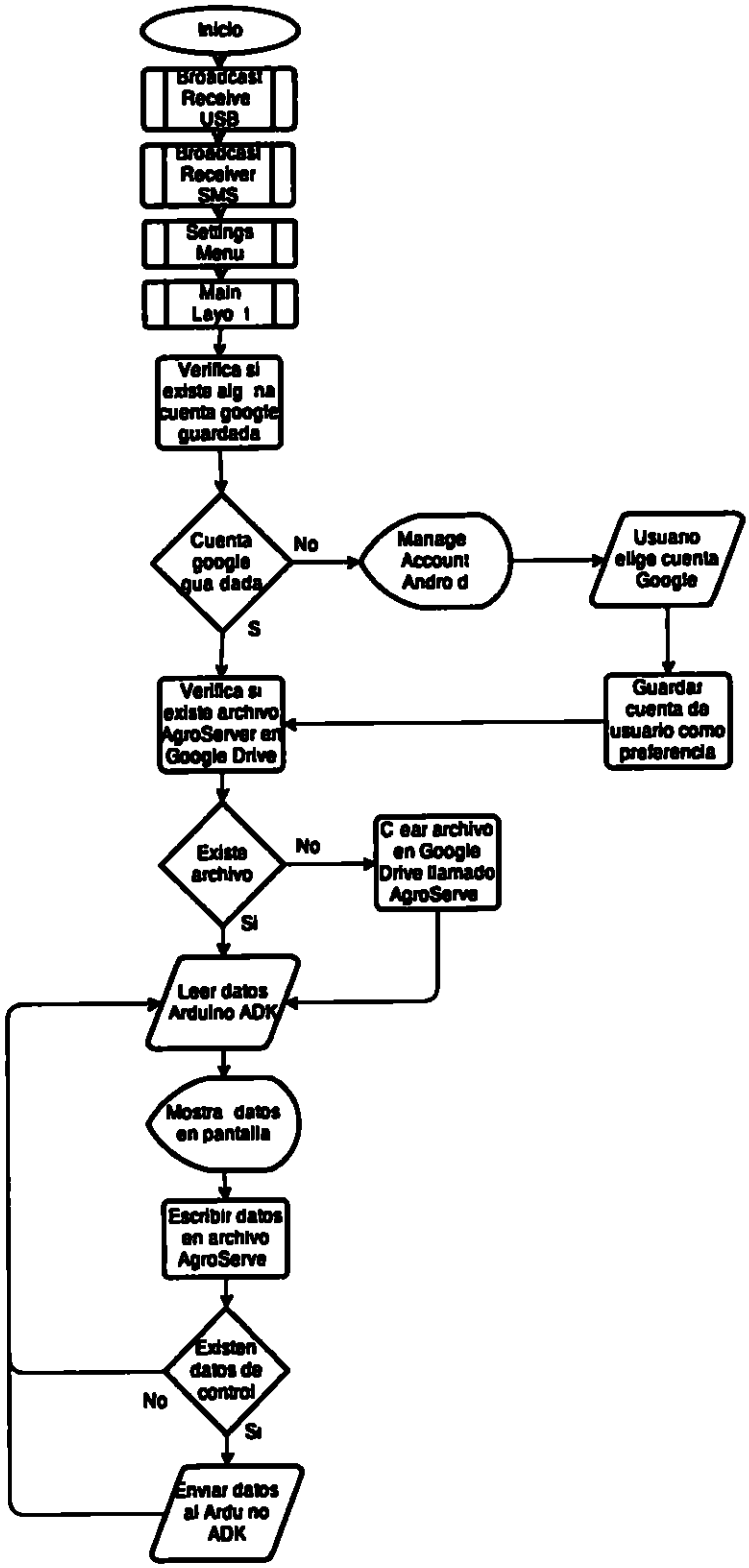


Fig 37 Diagrama de flujo general de AgroServer

3 2 2 AgroUser

Es la aplicación que utilizará el agricultor para verificar y controlar remotamente el cultivo Los objetivos que cumple la aplicación se mencionan a continuación

Monitorear los datos del cultivo almacenados en Google Drive

Controlar el sistema remotamente por medio de SMS

Al abrir la aplicación por primera vez en el dispositivo Android la misma le solicitará un numero telefónico el cual debe corresponder con el numero telefónico del dispositivo donde se encuentra instalada la aplicación AgroServer Este numero telefónico nos servirá para enviar los SMS de control al sistema de riego Una vez agregado el numero telefónico el mismo queda almacenado en las preferencias de la aplicación y el agricultor no tendrá que pasar nuevamente por este proceso en las próximas ejecuciones de la aplicación

La pantalla principal de la aplicación mostrara la página de inicio de Google Drive tres botones idénticos en apariencia y funcionalidad a la aplicación AgroServer pero con la diferencia de que el dato de control va dentro de un SMS

La decisión de elegir SMS (Short Message Service) para enviar los datos de control se debe principalmente a su facilidad y viabilidad de implementación El envió y la recepción de un SMS es prácticamente instantáneo debido a que son manejados solamente a nivel de la red celular y con ayuda de su pequeño tamaño se transmiten rápidamente por estas razones desde sus inicios los mensajes de texto fueron utilizados en el sector industrial para enviar y recibir alarmas datos de sensores y todo tipo de datos además la comunicación entre máquinas mientras que en la actualidad la

tecnologia de transmisión celular mitad internet y mitad red celular no garantiza la velocidad en la entrega de datos si se desea adoptar el uso de Internet para transmitir y recibir información Otro punto a destacar de este servicio es su adopción en las nuevas tecnologias 4G de telefonia celular

A continuación en la (Fig 38) se muestra el diagrama de flujo de la aplicación AgroUser

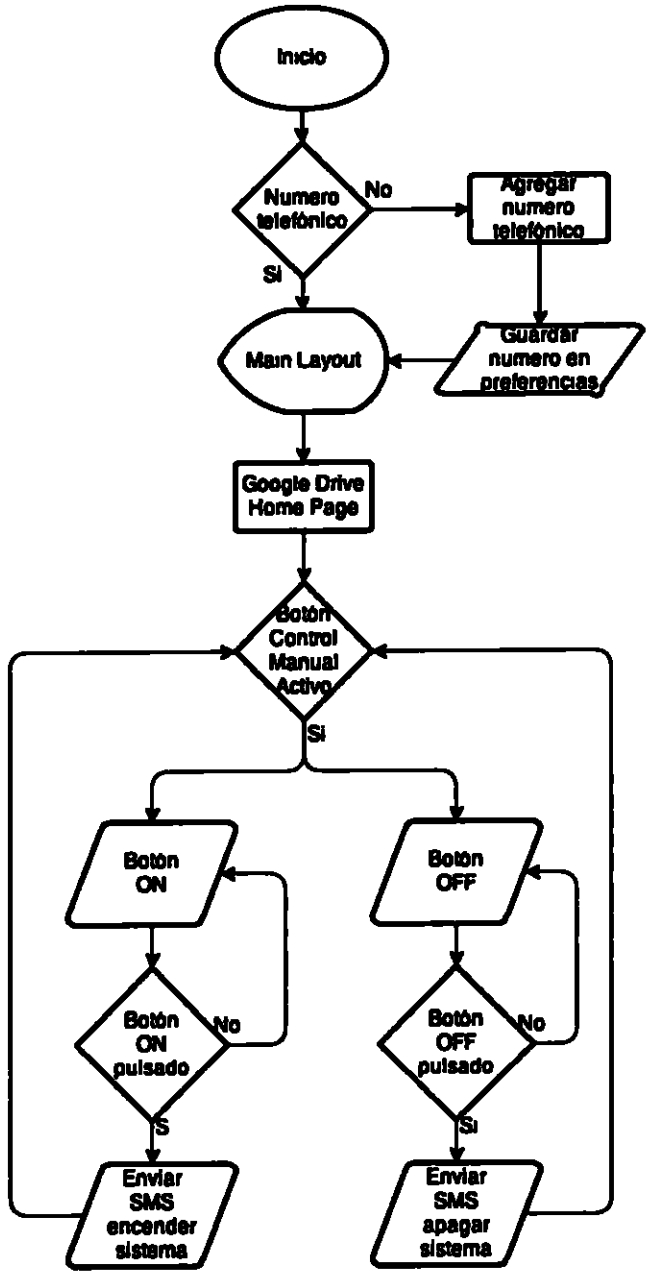


Fig 38 Diagrama de flujo AgroUser

3.3 Integración de las Partes

Logrando el funcionamiento estable de todas las partes (nodo principal nodo secundario y aplicaciones Android) de forma individual se procede a integrar dichas

partes para que se comuniquen entre si por medio de tecnologias como XBee USB Internet y redes celulares obteniendo un esquema de funcionamiento al mostrado en la (Fig 39)

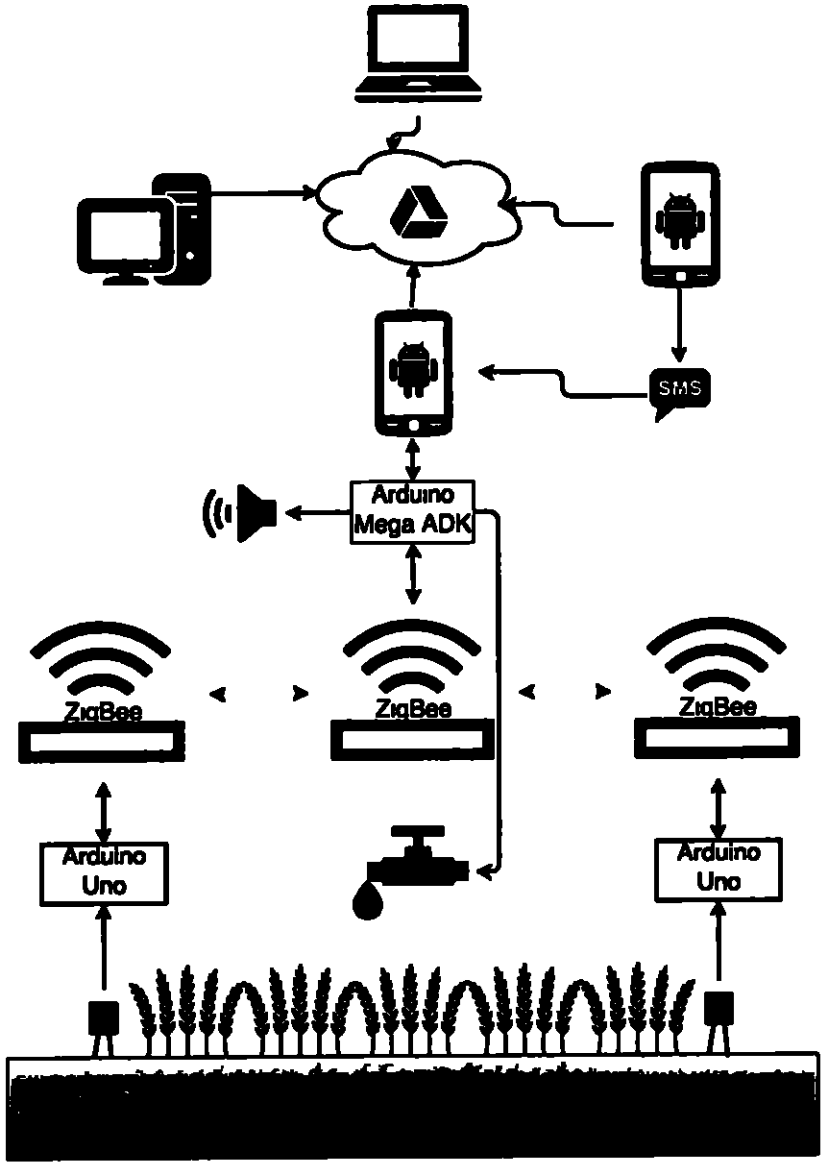


Fig 39 Esquema de funcionamiento del sistema

4 RESULTADOS Y DISCUSIONES

4 1 Requisitos de Operación

Para poder que funcione el prototipo son necesarias las siguientes prestaciones

Fuente de alimentación para cada nodo que sea capaz de entregar 7 a 12 voltios de corriente directa con 500 milis amperios

Smartphone o Tablet con sistema operativo Android con versión minima 4 0 (Ice Cream Sandwich) y que tenga conectividad a la red celular El mismo se utilizará en el nodo principal y tendrá instalado la aplicación AgroServer

Smartphone o Tablet con sistema operativo Android con versión minima 2 3 4 (Gingerbread) El mismo sirve para instalar la aplicación AgroUser

Conectividad a la red celular en el area de operacion de los nodos

4 2 Prueba de Captura de Datos

Es difícil comparar un sistema de riego controlado manualmente con uno automatizado Existen variables que el agricultor no puede medir para de esta forma poder ajustar el tiempo de riego y compensar la pérdida de agua algunas de ellas son la cantidad de agua que las plantas consumen (depende del tipo de planta y las etapas de desarrollo de la planta) radiación solar viento temperatura y humedad ambiente Debido a estas variables es difícil mantener un porcentaje de humedad estable con un sistema de riego controlado manualmente y si a ello le sumamos los errores humanos el cultivo no tendrá el mejor desarrollo produciéndole pérdidas directas al agricultor en su cosecha

Con el prototipo de sistema de riego automatizado que desarrollamos en esta investigación todas estas variables mencionadas anteriormente son irrelevantes debido a que se mide directamente la humedad del suelo para de esta forma controlar el encendido y apagado del riego así se mantiene una humedad promedio en el suelo y agua disponible siempre que la planta lo necesite En la (Fig 40) se muestra la gráfica del porcentaje de humedad promedio que mantiene nuestro prototipo

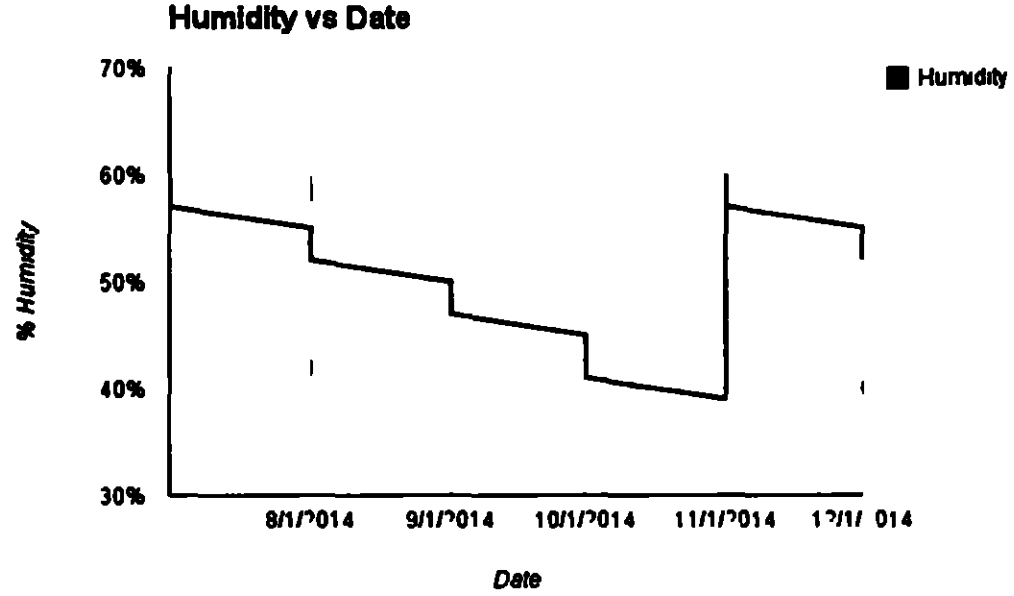


Fig 40 Humedad promedio que mantiene el suelo con el sistema automatizado

La prueba que nos permitió arrojar la gráfica que se muestra en la (Fig 40) fue realizada durante ocho dias continuos iniciando las pruebas cada día a las 8 00 am y finalizando a las 8 00 pm (19 00 horas) horas de Panamá La muestra de suelo utilizada fue de tipo franco arcilloso comunmente encontrada en áreas de toda la superficie de Panamá El porcentaje de humedad fue capturado por el sistema y alamcenado en el archivo AgroServer alojado en Google Drive cada hora de los diferentes dias Los datos capturados por dia los muestro a continuación en una serie de figuras

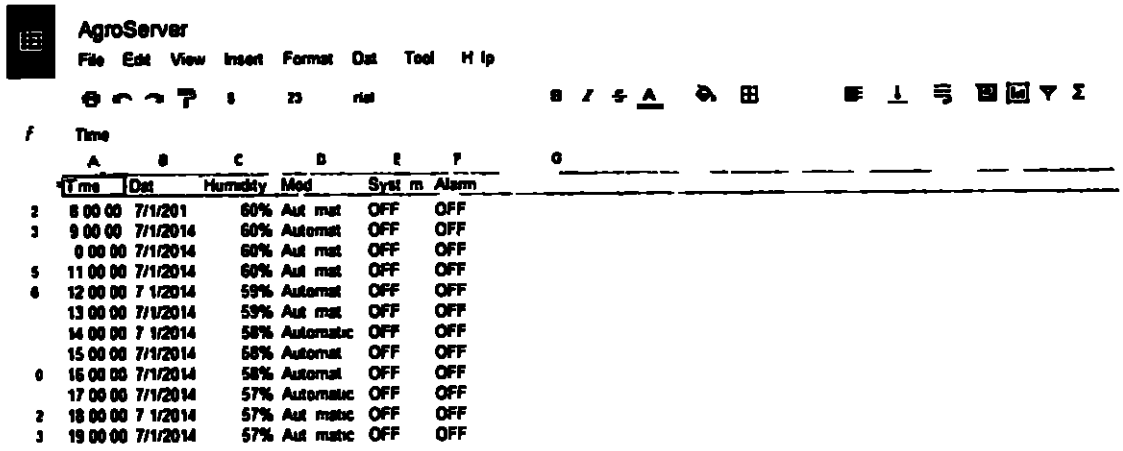


Fig 41 Datos capturados dia 1

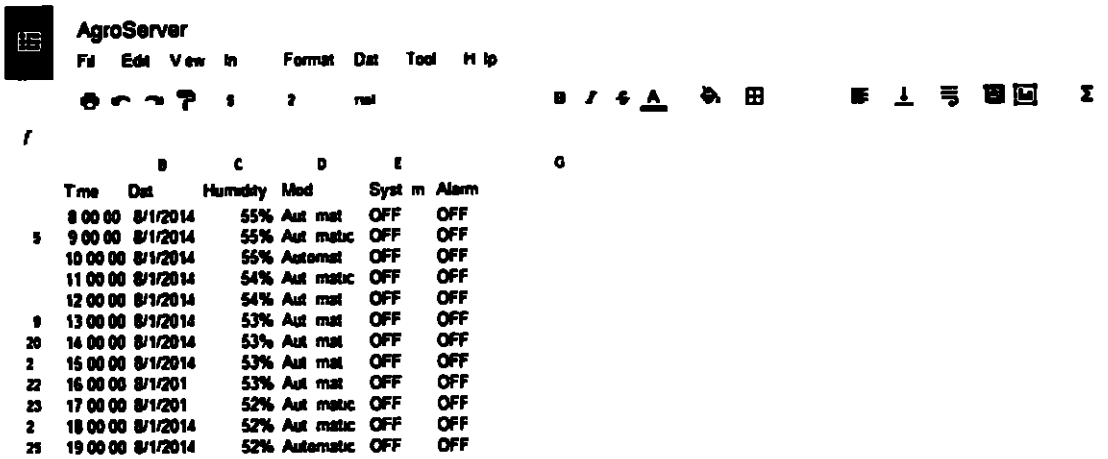


Fig 42 Datos capturados dia 2

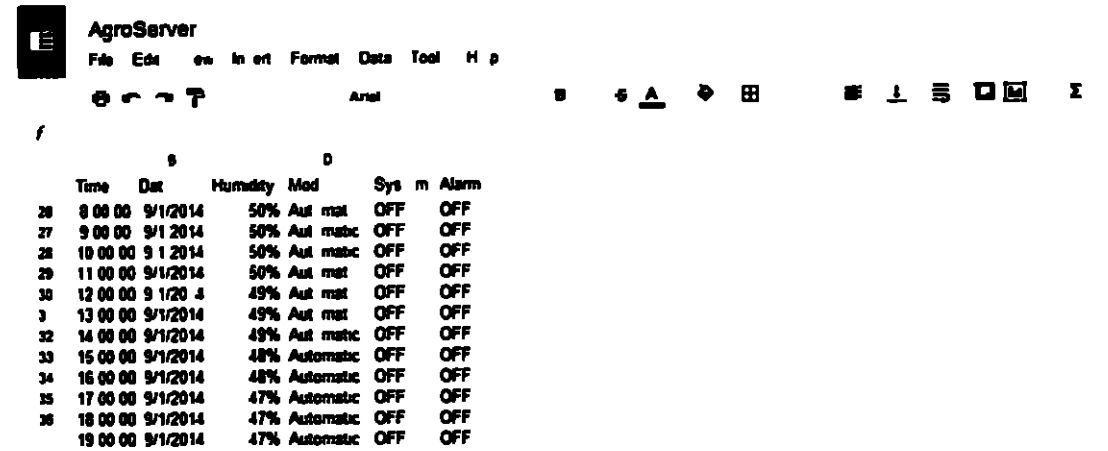


Fig 43 Datos capturados dia 3

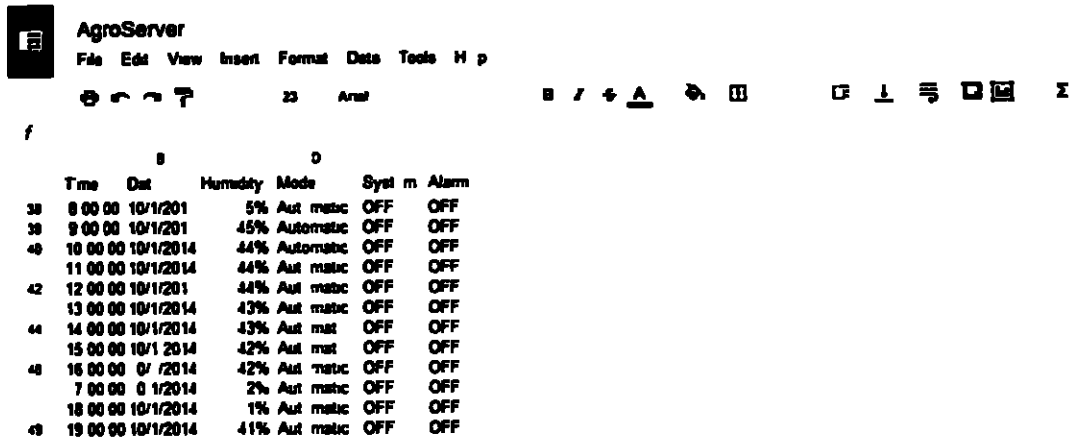


Fig 44 Datos capturados dia 4

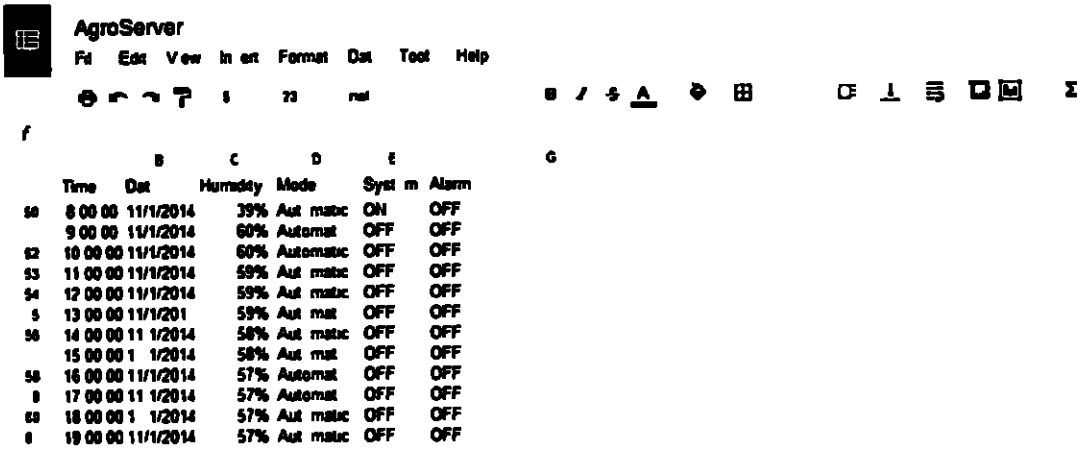


Fig 45 Datos capturados dia 5

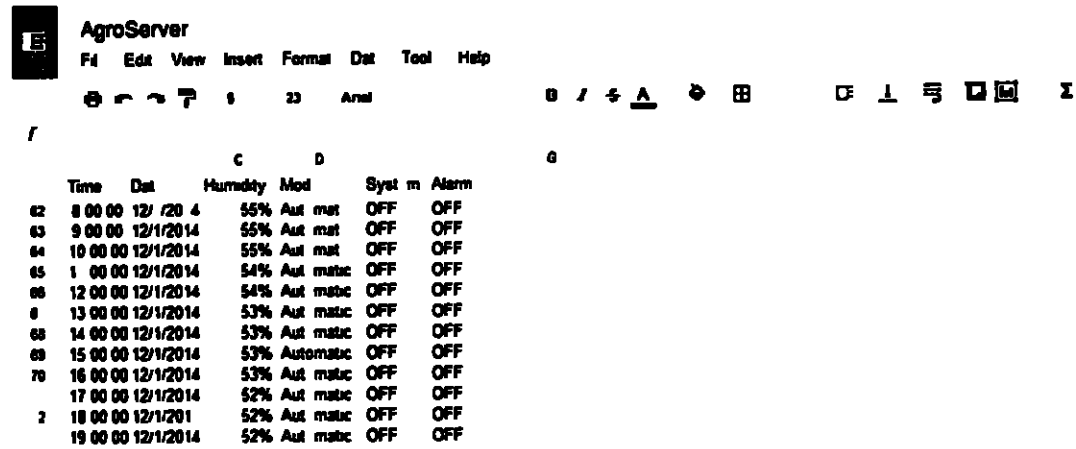
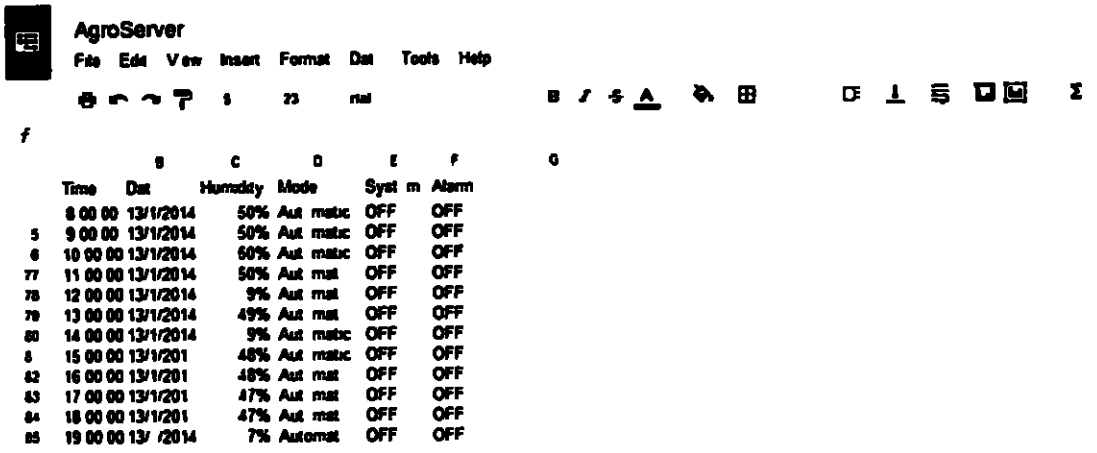


Fig 46 Datos capturados dia 6

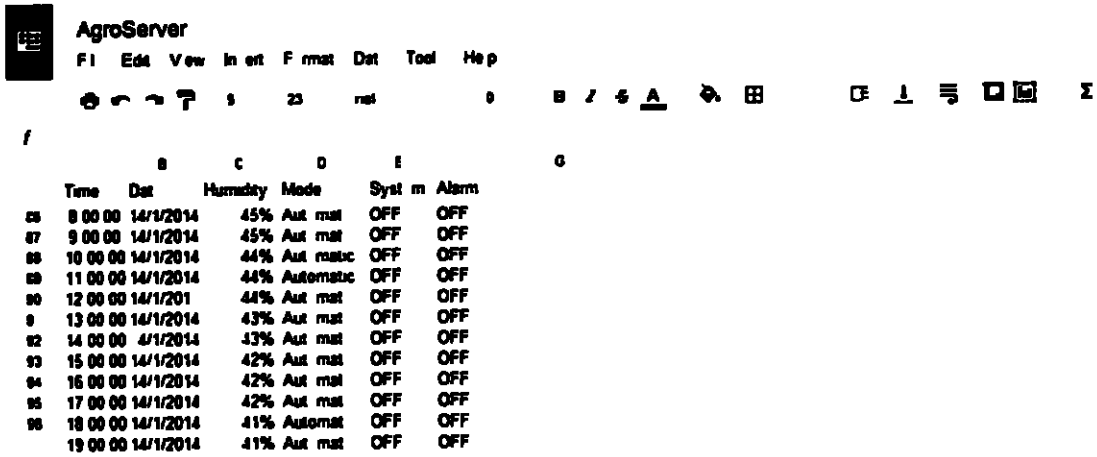


AgroServer

File Edit View Insert Format Dat Tools Help

	B	C	D	E	F	G
	Time	Dat	Humidity	Mode	Syst m	Alarm
	8 00 00	13/1/2014	50%	Aut matic	OFF	OFF
5	9 00 00	13/1/2014	50%	Aut matic	OFF	OFF
6	10 00 00	13/1/2014	50%	Aut matic	OFF	OFF
77	11 00 00	13/1/2014	50%	Aut mat	OFF	OFF
78	12 00 00	13/1/2014	9%	Aut mat	OFF	OFF
79	13 00 00	13/1/2014	49%	Aut mat	OFF	OFF
80	14 00 00	13/1/2014	9%	Aut matic	OFF	OFF
8	15 00 00	13/1/201	48%	Aut matic	OFF	OFF
82	16 00 00	13/1/201	48%	Aut mat	OFF	OFF
83	17 00 00	13/1/201	47%	Aut mat	OFF	OFF
84	18 00 00	13/1/201	47%	Aut mat	OFF	OFF
85	19 00 00	13/ /2014	7%	Automat	OFF	OFF

Fig 47 Datos capturados día 7



AgroServer

File Edit View Insert Format Dat Tool Help

	B	C	D	E	F	G
	Time	Dat	Humidity	Mode	Syst m	Alarm
86	8 00 00	14/1/2014	45%	Aut mat	OFF	OFF
87	9 00 00	14/1/2014	45%	Aut mat	OFF	OFF
88	10 00 00	14/1/2014	44%	Aut matic	OFF	OFF
89	11 00 00	14/1/2014	44%	Automatic	OFF	OFF
90	12 00 00	14/1/201	44%	Aut mat	OFF	OFF
9	13 00 00	14/1/2014	43%	Aut mat	OFF	OFF
92	14 00 00	4/1/2014	43%	Aut mat	OFF	OFF
93	15 00 00	14/1/2014	42%	Aut mat	OFF	OFF
94	16 00 00	14/1/2014	42%	Aut mat	OFF	OFF
95	17 00 00	14/1/2014	42%	Aut mat	OFF	OFF
96	18 00 00	14/1/2014	41%	Automat	OFF	OFF
	19 00 00	14/1/2014	41%	Aut mat	OFF	OFF

Fig 48 Datos capturados día 8

En la (Fig 45) se puede observar el encendido del sistema luego de bajar el porcentaje de humedad por debajo del 40% una vez el porcentaje de humedad sea igual o mayor a un 60% el sistema se apagara, de esta forma mantenemos un promedio de humedad en el suelo entre un 40% a 60% Cabe recordar que esta variable de humedad de referencia se debe ajustar a cada cultivo como se explicó en la sección metodológica de la investigación

Esta prueba fue realizada en varias ocasiones arrojando datos similares destacando la estabilidad del porcentaje de humedad de la muestra de suelo gracias al sistema automatizado Las tablas mostradas en la serie de figuras presentadas anteriormente fueron tomadas directamente del archivo de escritura de la aplicación AgroServer al igual que la gráfica mostrada en la (Fig 40 pág 89) esto demuestra la facilidad y versatilidad de este prototipo

4 3 Funcionamiento de la Aplicación AgroServer

Las pruebas realizadas a la aplicación consistieron en observar los datos capturados provenientes del Arduino Mega ADK para luego ser mostrados en pantalla operación de los botones de control manual creación del documento y escritura en el mismo La apariencia de la aplicación en funcionamiento se muestra en la siguiente figura

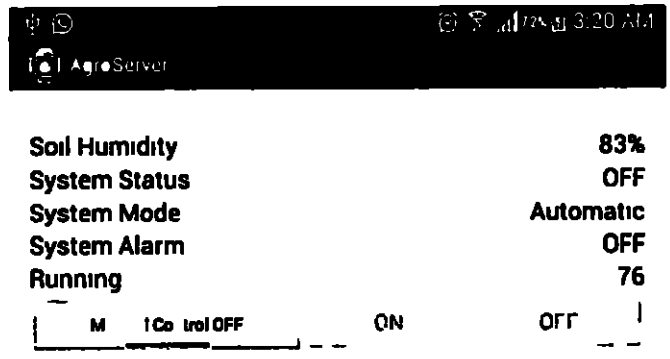


Fig 49 Apariencia general de la aplicación AgroServer en funcionamiento

Se puede observar en la figura el porcentaje de humedad el estado del sistema (en este caso apagado OFF debido al porcentaje de humedad) modo del sistema (por defecto el sistema inicia en modo automático) estado de la alarma (si el sistema está apagado no es necesario detectar el flujo de agua del sistema y por lo tanto el modo de la alarma se coloca en OFF) y su tres botones para el control manual Al principio de las pruebas se

contempló utilizar una variable llamada *Running*' que se muestra en la (Fig 50) para de esta forma observar la cantidad de ciclos de ejecución de la aplicación pero la versión final no contempla dicha variable

Si es activado el control manual del sistema a través de los botones la apariencia de la aplicación cambia un poco habilitando los dos botones que se observan sombreados en la (Fig 49 pág 93) La (Fig 50) muestra lo comentado

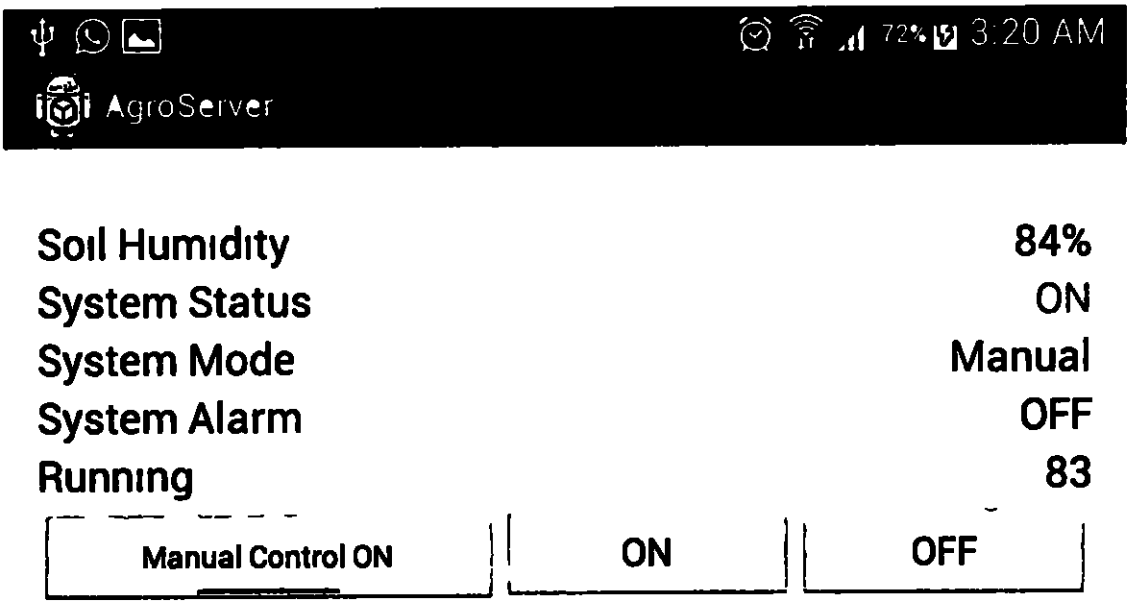


Fig 50 Modo automático del sistema

En la (Fig 50 pag 94) se presionó el botón Manual Control activando de esta forma el control manual del sistema y luego se presionó el botón ON para activar el sistema independientemente de la humedad detectada El control manual fuerza al sistema a encender o apagar dependiendo del botón presionado por el usuario

El menu de opciones que se ejecuta al presionar el botón de ajustes del dispositivo Android muestra dos opciones que se pueden observar en la siguiente figura

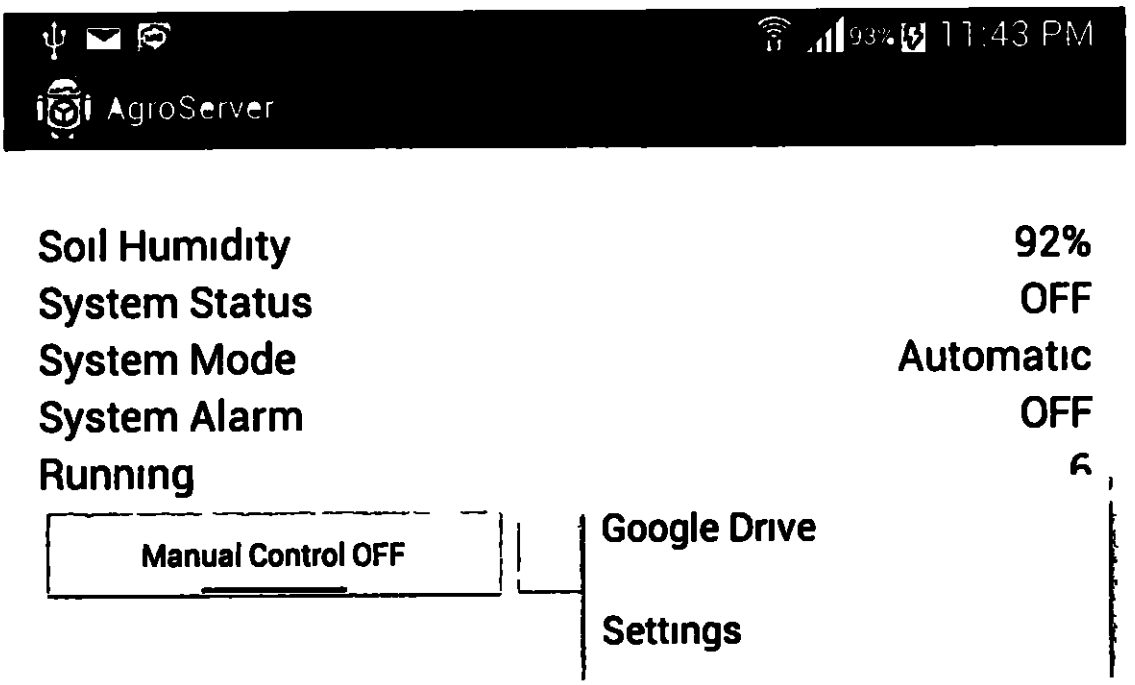


Fig 51 Menu de opciones

La opción *Google Drive* nos envía a una nueva ventana que nos muestra la página de inicio de Google Drive para luego de ingresar los datos poder observar el documento *AgroServer* donde se encuentran los datos del sistema En la siguiente figura se puede observar la ventana comentada

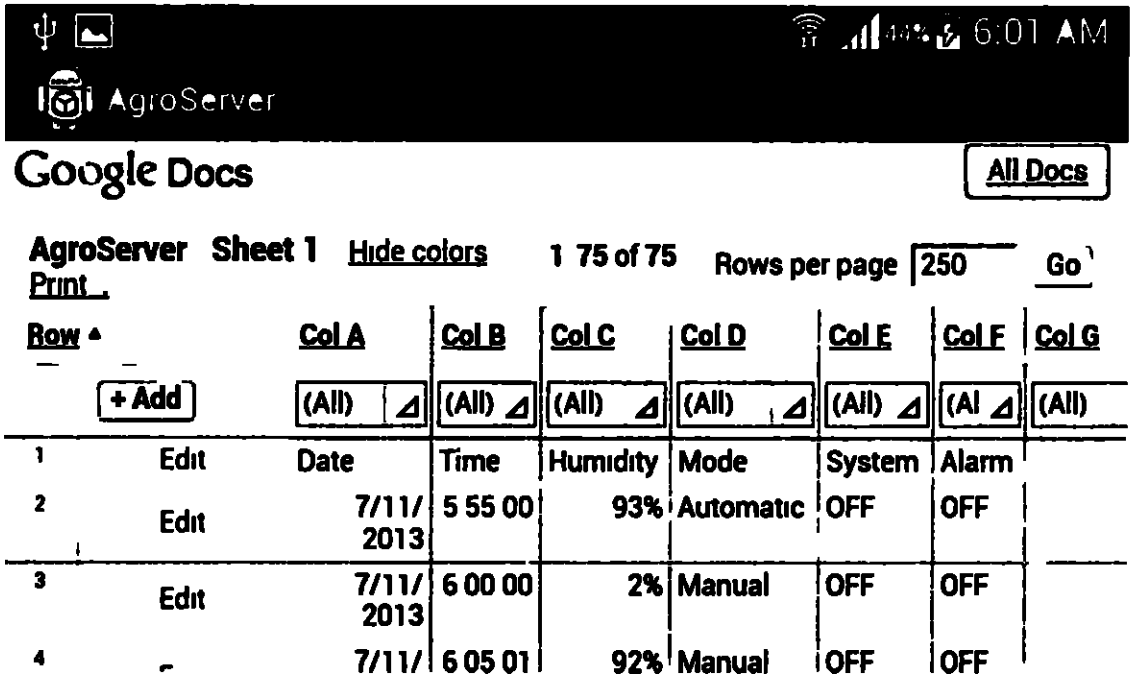


Fig 52 Google Drive

Si se elige la opción *Settings* mostrada en la (Fig 51 pág 95) se muestra el *Manager Account* del sistema operativo Android para vincular otra cuenta a la aplicación si así se desea Lo comentado se muestra en la siguiente figura

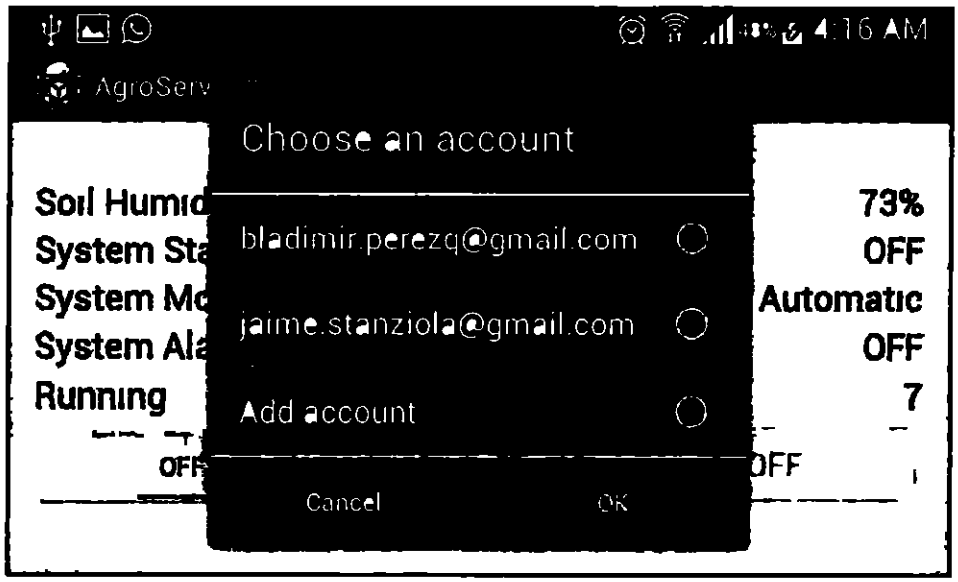


Fig 53 Manager Account Android

En el caso mostrado en la (Fig 53 pag 96) el dispositivo Android contaba con dos cuentas guardadas que se pueden vincular a la aplicación recordemos que la cuenta tiene que ser de Google

El código escrito en lenguaje de programación Java de la clase principal de la aplicación (MainActivity java) se puede observar en el Anexo C y el código de la clase secundaria (SmsReceiver java) que maneja los SMS entrantes se encuentra en el Anexo D Adicional existe una clase que maneja la ventana que muestra la página principal de Google Drive llamada Drivegoogle java y su código se puede observar en el Anexo E

El archivo de permisos (AndroidManifest xml) de la aplicación que está escrito en lenguaje XML se puede observar en el Anexo F

4 4 Funcionamiento de la Aplicación AgroUser

Debido a que la aplicación AgroUser es menos compleja en funcionamiento que AgroServer las pruebas se enfocaron principalmente en el envió de los mensajes de control a través del servicio SMS La aplicación luce como en la siguiente figura

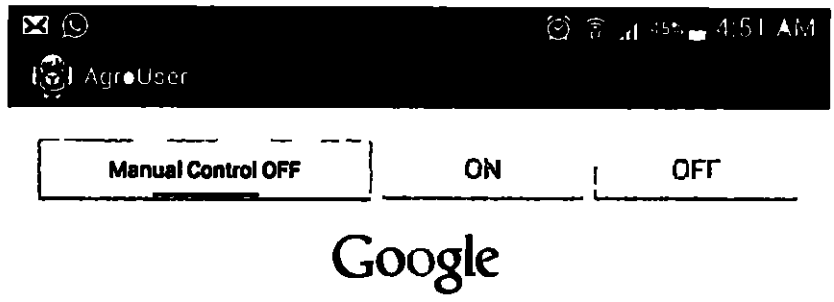


Fig 54 AgroUser

La aplicación muestra la página de inicio Google Drive para ingresar los datos de la cuenta y así poder observar el archivo *AgroServer'* que contiene los datos del sistema También contiene los tres botones con igual apariencia que en la aplicación AgroServer y su funcionamiento es bastante parecido con la diferencia de que los datos de control se envían por SMS Un detalle a resaltar es que cada vez que se presione uno de los botones (ON/OFF) se enviara un SMS de control para encender o apagar el sistema respectivamente La apariencia de la aplicación en modo manual se muestra en la siguiente figura

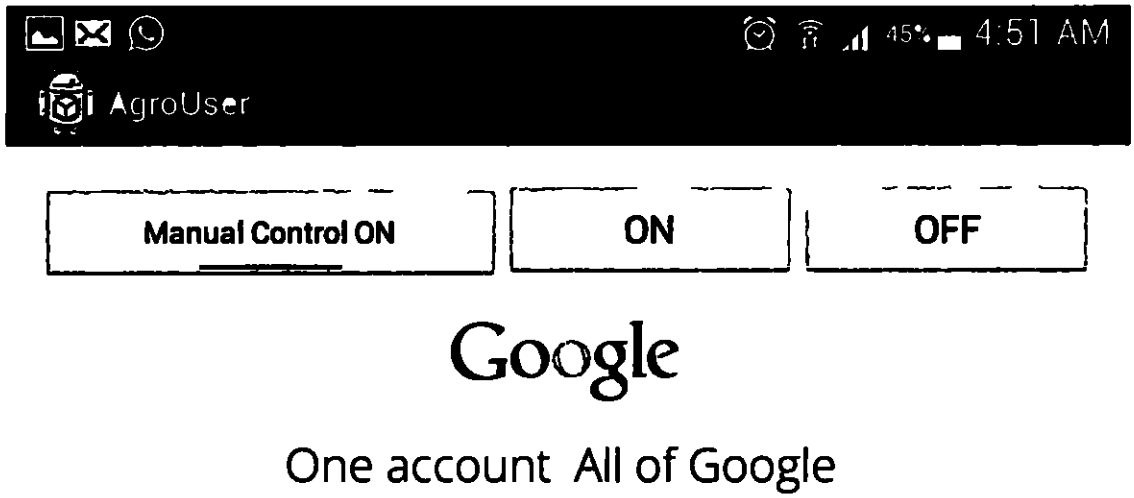


Fig 55 AgroUser Control Manual

Observamos que el botón llamado Manual Control tiene una pequeña franja que cambia de color gris a color celeste cuando está activo habilitando los dos siguientes botones para que el usuario pueda tomar la decisión de encender o apagar el sistema En la (Fig 56 pag 99) se puede observar el archivo *AgroServer'* dentro de la aplicación

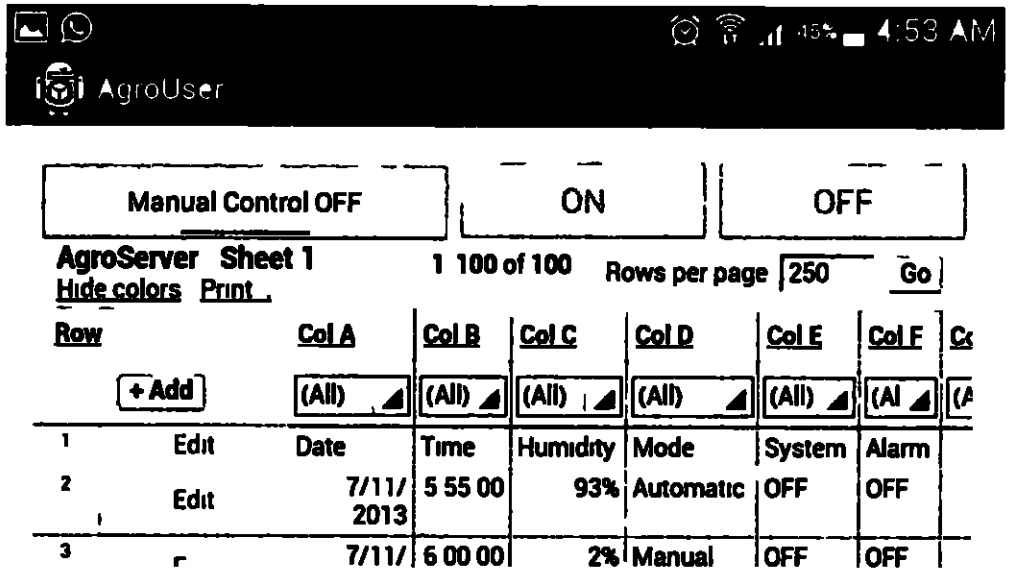


Fig 56 AgroUser mostrando el archivo de datos

La opción de cambiar el numero de destino de los SMS se muestra al presionar el botón de ajustes del dispositivo Android con nombre *Settings* presentara la ventana mostrada en la (Fig 57) para ingresar el nuevo numero vinculado al dispositivo que tiene instalada la aplicación AgroServer

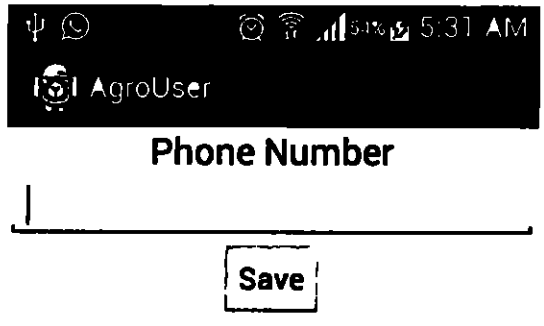


Fig 57 AgroUser editando el numero vinculado a AgroServer

Existen dos clases una llamada MainActivity.java que es la clase principal de la aplicación su código se encuentra en el Anexo G y otra llamada UserData.java que maneja el almacenamiento del numero vinculado a la aplicación AgroServer su código se observa en el Anexo H

4.5 Nodos en Operacion

Gracias a los nodos el sistema obtiene su autonomia, ya que a través de ellos obtenemos un valor de humedad que se encuentra en la vida real y se lleva al mundo digital para a través de él tomar una decisión En la siguiente figura se muestra la apariencia del nodo primario que utilizamos durante toda esta investigación

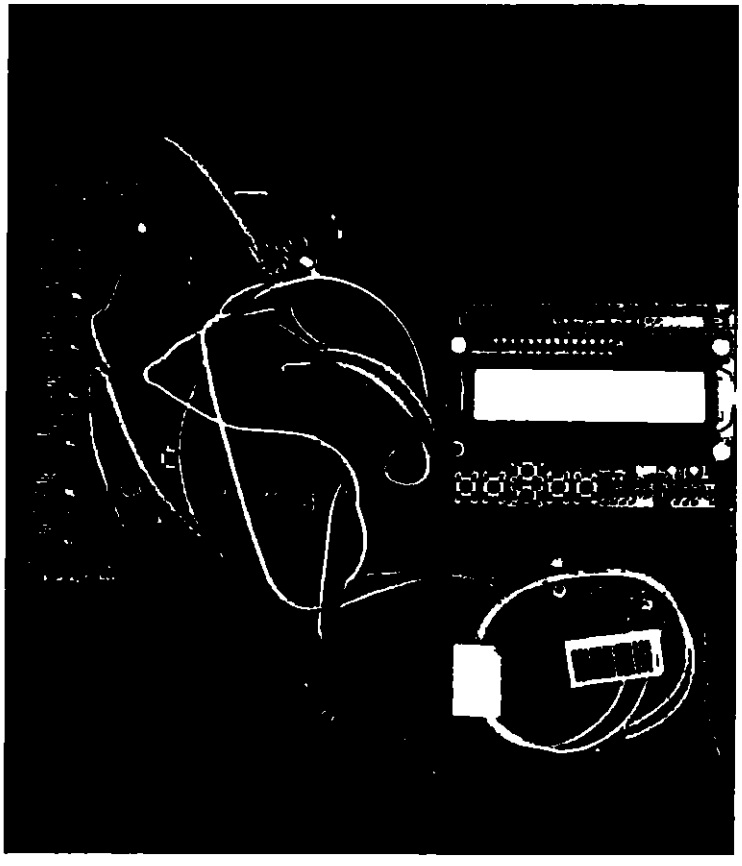


Fig 58 Nodo principal

El nodo principal consta de la tarjeta de relay display LCD 16x2 buzzer Arduino Mega ADK Wireless SD shield XBee serie 2 y una tarjeta de expansión en donde convergen todas las conexiones esta expansión es mostrada en el centro de la (Fig 58 pag 100) El Arduino Mega ADK y el XBee no se observan en la figura ya que se encuentran debajo de la tarjeta de expansión debido a que los board compatibles con Arduino se pueden conectar uno encima del otro formando una especie de edificio horizontal El display LCD muestra las variables del sistema de una manera particular debido al tamaño del mismo y a la cantidad de variables En la siguiente figura se observa como presenta el display las variables

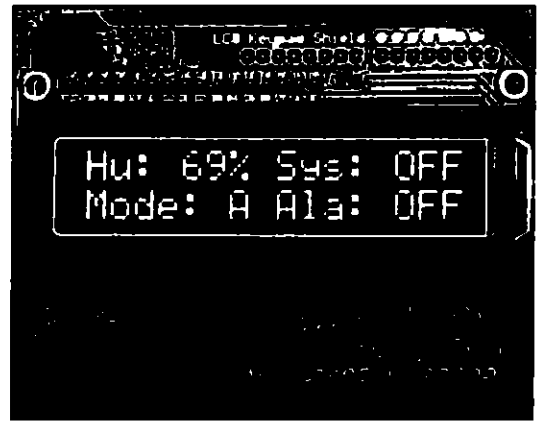


Fig 59 Funcionamiento del LCD 16x2

En la primera linea del LCD se observan dos valores llamados Hu y Sys estos corresponden a la humedad promedio y al estado del sistema (ON/OFF) respectivamente En la segunda linea los valores son Mode y Ala, los mismos corresponden al modo del sistema (A = automático M = manual) y Alarma (OFF/ON) Estos valores son los mismos que se envían a la aplicación AgroServer

El nodo secundario está formado por el Arduino Uno XBee serie 2 Wireless SD shield y soil moisture sensor En la siguiente figura se observa el nodo secundario

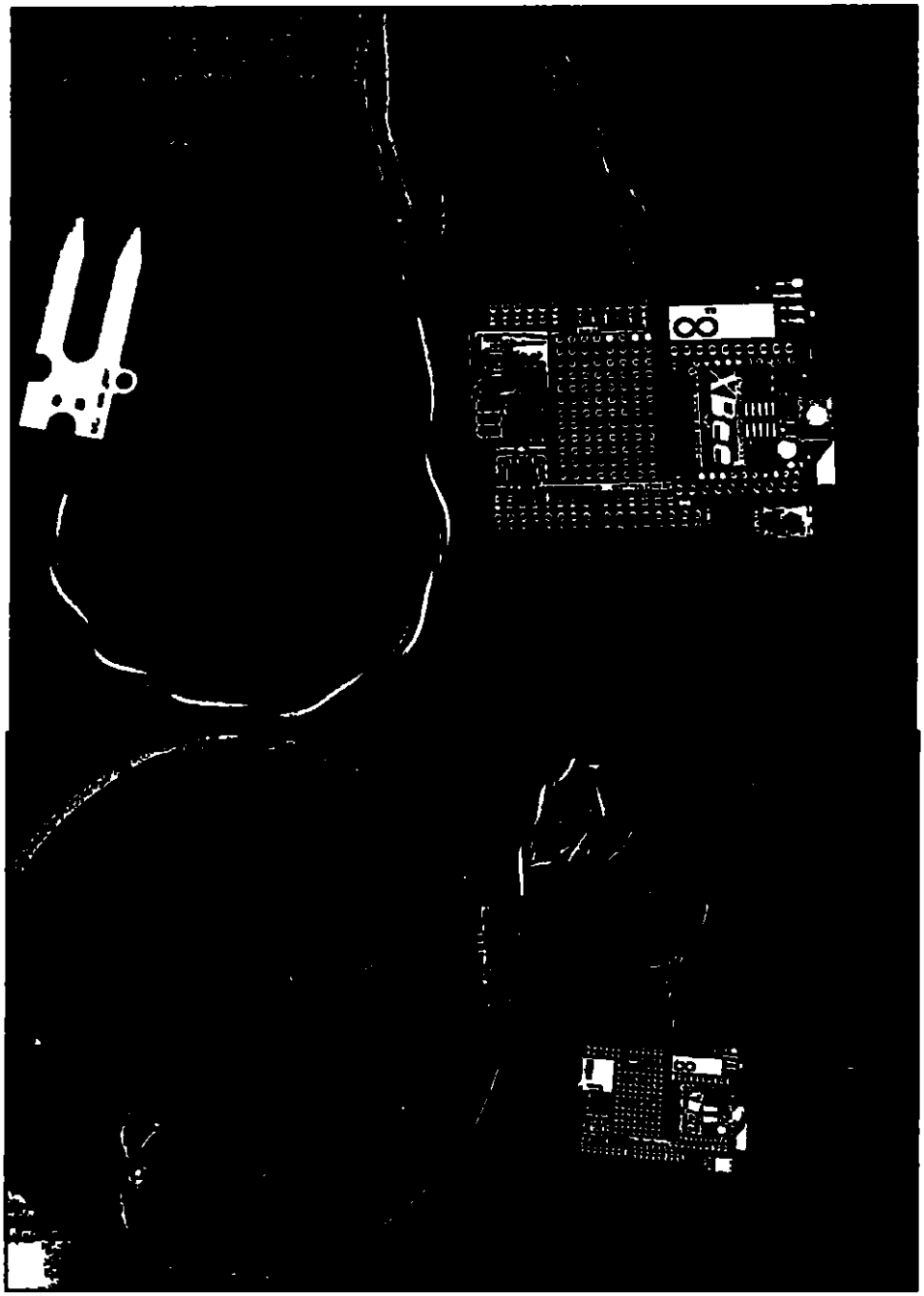


Fig 60 Nodo secundario

Al igual que en el nodo principal el Arduino Uno no se observa en la figura ya que se encuentra debajo del Wireless SD shield

El código del sketch del nodo principal se encuentra en el Anexo A y el del nodo secundario en el Anexo B

4 6 Costo del Prototipo

Una de las principales ventajas del prototipo es su costo que se detalla a continuación

Cuadro 5 Costo del prototipo utilizado en esta investigación

Descripción	Cantidad	Precio Unitario	Total
Arduino Mega ADK	1	70 00	70 00
Arduino Uno	1	50 00	50 00
XBee serie 2	2	40 00	80 00
Wireless SD Shield	2	30 00	60 00
Soil moisture sensor	1	10 00	10 00
8 Relay Board	1	10 00	10 00
Sensor de flujo	1	10 00	10 00
LCD 16x2	1	8 00	8 00
Fuente de alimentación	1	20 00	20 00
Total			**318 00**

Cabe destacar que los Smartphone y Tablets son gastos ya asumidos por los usuarios ya que en la actualidad la mayoría de las personas poseen un Smartphone con servicio de llamadas e Internet Esto representa un ahorro automático de aproximadamente 300 00 dólares en el desarrollo e implementación

4 7 Aspectos Innovadores

Lo principal a destacar de esta investigación en cuanto al aspecto innovador es lo siguiente

Herramientas de software y hardware con acceso libre a sus códigos fuentes como lo son Arduino y Android

Bajo costo de desarrollo

Ahorro del recurso hídrico del país gracias a la automatización del sistema

Monitoreo y control remoto del sistema

Utilización de nuevas tecnologías y flexibles a nuevos desarrollos Android Arduino y XBee

Leguajes de programación ampliamente utilizados Java y C++

Con lo anterior logramos brindarle al agricultor la comodidad ahorro y flexibilidad que obtenemos de un sistema automatizado

4 8 Problemas Presentados Durante el Desarrollo

4 8 1 Conexión Entre Plataformas

Uno de los problemas principales ocurridos en el desarrollo fue la poca información sobre la conectividad de Android con Arduino Mega ADK Es cierto que Android cuenta con la página oficial para comenzar a desarrollar con ADK e inclusive una aplicación demo publicada en su tienda Google play pero esto fue una de las primeras pruebas que realice y nunca nos funcionó dicho demo Por lo tanto decidimos comenzar por el

principio y enfocarnos en la página oficial de Android ADK (http //developer android com/tools/adk/index html) documentándonos de sus librerías y la guía 2012 publicada en dicha página, logrando así la comunicación del Arduino Mega ADK con dispositivos Android

4 8 2 Compatibilidad de Dispositivos Android

Existe un detalle en la conexión entre plataformas que no es garantizado y se comenta en la página web antes mencionada, que trata de la compatibilidad entre plataformas Segun la página web los dispositivos con sistema operativo Android de las versiones 2 3 4 y 3 0 o más recientes son compatibles para la conexión entre Arduino Mega ADK y Android pero solo aseguran dicha conexión para los dispositivos modelos Nexus diseñados por la empresa Google Teniendo esta información se procedió a intentar lograr la conexión con diferentes Smartphone y Tablets logrando la conexión en algunos de ellos que se comentan a continuación

Acer Iconia A500 con sistema operativo versión 4 0 3

Samsung Google Nexus S I9020T con sistema operativo versión 4 1 2

Samsung I9100 Galaxy S II con sistema operativo Android personalizado llamado CyanogenMod versión 10 1 y versión de Android 4 2 2 También se realizó la prueba con versión de Android 4 0 3 4 0 4 y 4 1 2 La mayoria de las pruebas se realizaron en este modelo de Smartphone debido a que es de uso personal

Samsung Galaxy Note II N7100 con sistema operativo Android versión 4 1 2

Si observamos en lo comentado anteriormente no logramos una conexión entre dispositivos Android con versiones por debajo de la 4 0 durante nuestra investigación

4 8 3 AgroServer

La investigación tomó mucho tiempo en el desarrollo de dicha aplicación debido al numero de servicios que debe manejar permanentemente

Una vez que se logró la conexión con el Arduino Mega ADK y el manejo de los paquetes de datos que se envían se procedió a desarrollar las rutinas necesarias para escribir dichos datos en un archivo Google Spreadsheet con nombre AgroServer que es en la actualidad el usado para dicha labor logrando este paso obteníamos el monitoreo del sistema en cualquier parte del mundo donde tengamos acceso a Internet y a nuestra cuenta Google Drive El problema radicaba en que la unica forma encontrada en esos entonces era crear el archivo manualmente en la cuenta Google Drive para luego estar disponible para su uso en la aplicación AgroServer Esta opción funcionaba correctamente y era simple a la hora de escribir los datos en el archivo ya que a la aplicación solo se le añadía un servicio más para poder acceder a Internet y no interfería con la recepción y transmisión de datos con el Arduino Mega ADK Esta opción no era práctica para el usuario y desde ese momento pensamos en la idea que la aplicación creara el archivo automáticamente y escribiera en él Esta idea ocasionaba otros problemas debido a que necesitariamos utilizar un servicio adicional llamado Google Drive SDK que contiene todas las herramientas para integrar la nube de Google Drive con nuestra aplicación Por lo tanto era necesario seguir los pasos para instalar este nuevo servicio en nuestro entorno de programación Eclipse y probar la serie de librerias documentos y ejemplos que tiene la página oficial para luego adaptarlo a lo que deseábamos ya que no se encuentra un ejemplo trabajando juntos Google Drive

Spreadsheet y Android Google Drive es un servicio muy amplio haciendo las cosas no tan sencillas debido a que en este caso es necesario utilizar una credencial para ingresar a la cuenta Google Drive del usuario y poder manipular sus archivos por ello en nuestra aplicación se utiliza el uso del *Account Manager*' del sistema operativo Android nuestra aplicación no maneja directamente una contraseña de usuario sino una ficha o Token que es creado a la hora de elegir una cuenta de usuario Eligiendo la cuenta la aplicación solicita a Google un permiso y si la cuenta es válida Google envía un Token que en adelante utilizará la aplicación para acceder a Google Drive y a cualquier otro servicio relacionado con Google Para lograr trabajar con Google Drive necesitamos desarrollar una aplicación aparte de la investigación para entender bien el funcionamiento de dicho servicio para luego acoplarlo a la comunicación con Arduino Mega ADK Ahora bien solo con Google Drive estábamos manejando tres servicios solicitud de Token creación de archivo AgroServer y escritura del mismo Manejar tantos servicios en una misma aplicación Android puede ocasionar una serie de errores si son ejecutados a la fuerza por el desarrollador y peor si trabajan en paralelo Toda aplicación Android maneja un proceso o hilo por defecto si dicho hilo demora ejecutando una instrucción más de 5 segundos el sistema operativo detiene toda la aplicación indicando un error Esto era un problema directo para nosotros ya que todos los servicios mencionados anteriormente pueden tomar más de 5 segundos de ejecución y si los forzaba a trabajar la aplicación nunca iba a trabajar Para ello es necesario utilizar hilos independientes dentro de la aplicación para que así el sistema operativo tome el control de dichos hilos y los ejecute en el tiempo considerado evitando el colapso de la aplicación es decir en nuestra

aplicación AgroServer existen una serie de hilos trabajando ocultos al usuario que realizan cada uno de estos servicios

CONCLUSIONES

Con esta investigación se logró diseñar un prototipo capaz de controlar un sistema de riego agrícola y obtener un funcionamiento autónomo a bajo costo

El sistema tomará la decisión de encender o apagar el riego cuando el porcentaje de humedad del suelo no esté dentro de los parámetros del cultivo garantizando un ahorro en el recurso hídrico y energético del sistema, disminuyendo los costes de producción al agricultor

El agricultor puede monitorear el estado del sistema remotamente con su Smartphone o Tablet gracias a la utilización de almacenamiento en la nube de Google Drive y el sistema operativo de código abierto Android Además puede controlarlo remotamente por medio de su aplicación móvil y servicios de SMS

La utilización de herramientas de software y hardware libre como Android y Arduino nos brindan la posibilidad de investigar soluciones futuras expandiendo la utilidad de esta investigación

Con este sistema autónomo el agricultor no tendrá que estar pendiente del riego del cultivo como normalmente solía hacerlo brindándole comodidad flexibilidad y seguridad que su cultivo siempre tendrá el agua necesaria para su desarrollo

TRABAJOS FUTUROS

A continuación comentare varias ideas que pueden ser el futuro de nuestra investigación

Diseñar una red de sensores que pueda cubrir una siembra agrícola independientemente de su tamaño por medio de redes Mesh

Desarrollar un sistema que analice las etapas del cultivo para proporcionarle el abono necesario en cada etapa

Investigar la forma de automatizar la red de válvulas de un cultivo para controlar el riego por parcelas independientemente del tamaño de la red

Utilizar analítica de video para por ejemplo verificar crecimiento del cultivo detectar plagas en el cultivo malezas entre el cultivo riego irregular en el cultivo

En la actualidad existen equipos llamados Drones que son capaces de volar bucear y andar sobre terrenos difíciles además contienen cámaras y sensores para desarrollar soluciones Con estos equipos se podría fumigar el cultivo si se requiere además de realizar una especie de patrullaje periódicamente para verificar alguna anomalía

Logrando cada una de estas ideas podríamos tener un sistema totalmente autónomo y una ganancia en la cosecha de casi un 100%

BIBLIOGRAFIA

Accessory Development Kit Disponible en http //developer android com/tools/adk/index html (Consultado a 15 10 2012 Actualizado a 29 12 2013)

Böhmer Mario 2012 Beginning Android ADK with Arduino New York Estados Unidos

Digi International 2008 XBee/Xbee PRO ZB OEM RF Modules

Edward C Martin 2010 Métodos para medir la humedad del suelo para la programación del riego ¿cuándo? Colegio de agricultura y ciencias de vida de la Universidad de Arizona

Glaria, Jaime Kouro Samir 2001 Sensores de humedad Universidad técnica Federico Santa María

Gutiérrez, Daniel Muñoz, Paul Suarez Arturo Automatización de un Sistema de Riego Agricola por Técnica de Goteo y Aspersión

Montesinos José 2013 Red de sensores auto configurable mediante tecnologias ZigBee y Arduino con monitorización por aplicación Android Tesis Universidad Politécnica de Cartagena, Cartagena, Colombia

Oyarce Andrés 2010 Guia del Usuario XBee Series 1

Perera, Achala 2010 ZigBee Wireless Soil Moisture Sensor Design for Vineyard Management System Tesis Auckland University of Technology 113 Págs

Ramírez, Jorge Jara Valenzuela Avilés Alejandro 1998 Desarrollo de Sistemas de Riego en el Secano Interior y Costero Universidad de Concepción Campus Chillán Chile

Schugurensky Carlos Capraro Flavio Control Automático de Riego Agrícola con Sensores Capacitivos de Humedad de Suelo Aplicaciones en Vid y Olivo

Torres Oscar 2013 Arduino curso práctico de formación

'xbee arduino Disponible en https //code google com/p/xbee arduino (Consultado a 07 05 2013 Actualizado a 05 01 2014)

ANEXOS

Anexo A Sketch nodo primario

```
#include <Max3421e h>
#include <Usb h>
#include <AndroidAccessory h>
#include <LiquidCrystal h>
#include <XBee h>

#define RELAY1_PIN 12
#define ALARM_PIN 8
#define BUZZER_PIN 9

XBee xbeeTx = XBee()
uint8_t orden[] = { 1 }

//SH and SL addresses
uint32_t sh1 = 0x0013a200
uint32_t sl1 = 0x4086b418
uint32_t sh[] = {sh1}
uint32_t sl[] = {sl1}

ZBTxStatusResponse txStatus = ZBTxStatusResponse()
ZBRxResponse rx = ZBRxResponse()

int delayAll = 500              //delay para controlar los ciclos del programa en general
int delayflow = 2000            //delay del sensor de flujo
int time = 60
int contador = 0
int len = 0
int flag2 = 0
int modems = 1
int humedadreferencia = 40
LiquidCrystal lcd(6 7 2 3 4 5)
byte lectura[0]
```

```
byte escritura[3]              //escritura[0] = humedad
                               //escritura[1] = system status = on(1)/off(0)
                               //escritura[2] = system mode = automatico(1)/manual(0)
                               //escritura[3] = alarma = on(1)/off(0)

AndroidAccessory      acc( Manufacturer        Model        Description        1 0
 http //yoursite com    0000000012345678 )

void setup()
{
 xbeeTx begin(9600)
 Serial begin(9600)
 lcd clear()
 lcd home()
 lcd begin(16  2)
 pinMode(RELAY1_PIN  OUTPUT)
 digitalWrite(RELAY1_PIN HIGH)
 delay(100)
 acc powerOn()
}

void loop()
{
 if (acc isConnected()) {
  len = acc read(lectura, sizeof(lectura)  1)
  switch(lectura[0]){
   case 0
    automatico()
      break
   case 10
    manualoff()
      break
   case 11
    manualon()
      break
  }
  escribe()
 }

 else{
  automatico()
 }
 printlcd()
}
```

```
void readhumidity(){
 int humidity = 0
 int i
 for(i = 0  i < modems  i++){
  humidity = senddata(sh[i] sl[i]) + humidity
 }
 humidity = humidity / modems
 escritura[0] = (byte)humidity
}

void systemon(){
 int sensorflow
 digitalWrite(RELAY1_PIN LOW)
 delay(delayflow)
 sensorflow = digitalRead(ALARM_PIN)
 if(sensorflow == 1){
  escritura[1] = (int)1
  alarmoff()
  humedadreferencia = 60
  return
 }
 else{
    systemoff()
    delay(delayflow)
    alarmon()
    return
 }
}

void systemoff(){

 digitalWrite(RELAY1_PIN HIGH)
 escritura[1] = (int)0
 alarmoff()
 humedadreferencia = 40

}

void automatico(){
 readhumidity()
 if((int)escritura[0] <= humedadreferencia){
  systemon()
 }
 else{
  systemoff()
```

```
 }
 escritura[2] = (int)0
}

void manualoff(){
 readhumidity()
 systemoff()
 escritura[2] = (int)1
}

void manualon(){
 readhumidity()
 systemon()
 escritura[2] = (int)1
}

void alarmon(){
 escritura[3] = (int)1
 tone(BUZZER_PIN 1900)
}

void alarmoff(){
 escritura[3] = (int)0
 noTone(BUZZER_PIN)
}

void printlcd(){
 lcd clear()
 lcd setCursor(0 0)
 lcd print( Hu   )
 lcd print(escritura[0])
 lcd print( % )
 switch(escritura[1]){
  case 1 {
   lcd setCursor(8 0)
   lcd print( Sys  ON )
   break
  }
  case 0 {
   lcd setCursor(8 0)
   lcd print( Sys  OFF )
   break
  }
 }
 switch(escritura[2]){
```

```
  case 1 {
   lcd setCursor(0 1)
   lcd print( Mode  M )
   break
  }
  case 0 {
   lcd setCursor(0 1)
   lcd print( Mode  A )
   break
  }
 }
 switch(escritura[3]){
  case 1 {
   lcd clear()
   lcd setCursor(0 0)
   lcd print( Humidity   )
   lcd print(escritura[0])
   lcd print( % )
   lcd setCursor(0 1)
   lcd print( Alarm  ON )
   break
  }
  case 0 {
   lcd setCursor(8 1)
   lcd print( Ala  OFF )
   break
  }
 }
}

int senddata(uint32_t sh  uint32_t sl){
 XBeeAddress64 addr64 = XBeeAddress64(sh  sl)
 ZBTxRequest zbTx = ZBTxRequest(addr64  orden  sizeof(orden))
 xbeeTx send(zbTx)
 if (xbeeTx readPacket(2000)) {
  if (xbeeTx getResponse() getApiId() == ZB_TX_STATUS_RESPONSE) {
   xbeeTx getResponse() getZBTxStatusResponse(txStatus)
   if (txStatus getDeliveryStatus() == SUCCESS) {
    return receivedata(sh  sl)
   } else {
    senddata(sh sl)
   }
  }
  else{
   senddata(sh sl)
```

```
    }
   }
   else if (xbeeTx getResponse() isError()) {
    senddata(sh sl)
   }
   else {
    senddata(sh sl)
   }
}

int receivedata(uint32_t sh  uint32_t sl){
 if (xbeeTx readPacket(2000)) {
   if (xbeeTx getResponse() isAvailable()) {
     if (xbeeTx getResponse() getApiId() == ZB_RX_RESPONSE) {
       xbeeTx getResponse() getZBRxResponse(rx)
       if((int)rx getData(0) <= 100 && (int)rx getData(1) == 5){
         return (int)rx getData(0)
       }
     }
     else {
       senddata(sh  sl)
     }
   }
   else{
     senddata(sh  sl)
   }
 }
 else if (xbeeTx getResponse() isError()) {
   senddata(sh  sl)
 }
}

void escribe(){

 acc write(escritura, 4)

}
```

Anexo B Sketch nodo secundario

```
#include <XBee.h>
#include <SoftwareSerial.h>
#define SENSOR 0

XBee xbeeRx = XBee();
uint8_t data[] = {0, 5};

// SH y SL del coordinador
uint32_t sh = 0x0013a200;
uint32_t sl = 0x4086b427;

XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

int humedad = 0;

void setup() {
 xbeeRx.begin(9600);
 Serial.begin(9600);
}

void loop() {
 readdata();
}

void readdata(){
 xbeeRx.readPacket();
 if (xbeeRx.getResponse().isAvailable()) {
  if (xbeeRx.getResponse().getApiId() == ZB_RX_RESPONSE) {
   xbeeRx.getResponse().getZBRxResponse(rx);
   if((int)rx.getData(0) == 1){
    humedad = analogRead(SENSOR);
    data[0] = (byte)(((long)humedad * (long)100)/(long)950);
    senddata();
   }
  }
 }
}

void senddata(){
```

```
XBeeAddress64 addr64 = XBeeAddress64(sh  sl)
ZBTxRequest zbTx = ZBTxRequest(addr64  data  sizeof(data))
xbeeRx send(zbTx)
if (xbeeRx readPacket(2000)) {
 if (xbeeRx getResponse() getApiId() == ZB_TX_STATUS_RESPONSE) {
  xbeeRx getResponse() getZBTxStatusResponse(txStatus)
  if (txStatus getDeliveryStatus() == SUCCESS) {
   return
  }
  else {
   readdata()
  }
 }
}
}
```

Anexo C MainActivity.java (AgroServer)

```
package com.example.agroserver

import java.io.FileDescriptor
import java.io.FileInputStream
import java.io.FileOutputStream
import java.io.IOException
import java.net.URL
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Date
import java.util.List
import com.google.android.gms.auth.GoogleAuthException
import com.google.api.client.extensions.android.http.AndroidHttp
import
com.google.api.client.googleapis.extensions.android.gms.auth.GoogleAccountCredential
import
com.google.api.client.googleapis.extensions.android.gms.auth.UserRecoverableAuthIOE
xception
import com.google.api.client.json.gson.GsonFactory
import com.google.api.services.drive.Drive
import com.google.api.services.drive.DriveScopes
import com.google.api.services.drive.model.File
import com.google.gdata.client.spreadsheet.SpreadsheetService
import com.google.gdata.data.spreadsheet.CellEntry
import com.google.gdata.data.spreadsheet.CellFeed
import com.google.gdata.data.spreadsheet.ListEntry
import com.google.gdata.data.spreadsheet.ListFeed
import com.google.gdata.data.spreadsheet.SpreadsheetEntry
import com.google.gdata.data.spreadsheet.SpreadsheetFeed
import com.google.gdata.data.spreadsheet.WorksheetEntry
import com.google.gdata.data.spreadsheet.WorksheetFeed
import com.google.gdata.util.ServiceException

import android.os.AsyncTask
import android.os.Bundle
import android.os.ParcelFileDescriptor
import android.accounts.AccountManager
import android.app.Activity
import android.app.PendingIntent
import android.content.BroadcastReceiver
import android.content.Context
```

```
import android content Intent
import android content IntentFilter
import android content SharedPreferences
import android graphics Color
import android util Log
import android view Menu
import android view MenuItem
import android view View
import android view View OnClickListener
import android widget Button
import android widget TextView
import android widget Toast
import android widget ToggleButton

import com android future usb UsbAccessory
import com android future usb UsbManager

public class MainActivity extends Activity {

        Intent a

        GoogleAccountCredential credential

        Drive service

        URL SPREADSHEET_FEED_URL = null

        String accountName
        String systemstatus =
        String systemmode =
        String systemalarm =
        public String idfile

        static final int REQUEST_ACCOUNT_PICKER = 1
        static final int REQUEST_AUTHORIZATION = 2

        SharedPreferences userdata

        private static final String TAG =  ArduinoAccessory

        private static final String ACTION_USB_PERMISSION =
com android example USB_PERMISSION

  private UsbManager mUsbManager
  private PendingIntent mPermissionIntent
```

```
    private boolean mPermissionRequestPending

    UsbAccessory mAccessory
    ParcelFileDescriptor mFileDescriptor
    FileInputStream mInputStream
    FileOutputStream mOutputStream

    TextView humidity alarm status mode

    Button bon boff
    ToggleButton bmode

    public byte[] leer = new byte[4]   // leer[0] = % humedad
                                                    // leer[1] = estatus del
sistema, 0 apagado  1 encendido
                                                    // leer[2] = modo del
sistema, 1 manual  0 automatico
                                                    // leer[3] = alarma del
sistema, 1 on  0 off

    public byte[] write = new byte[1] // write[0] = codigo del sms
    boolean bandera=true
    int control = 0
    public int flagaddrow = 0                       //Esta bandera controla el
agregar una fila al spreadsheet

    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context  Intent intent) {
            String action = intent getAction()
            if (ACTION_USB_PERMISSION equals(action)) {
                    synchronized (this) {
                            UsbAccessory accessory = UsbManager getAccessory(intent)
                            if
(intent getBooleanExtra(UsbManager EXTRA_PERMISSION_GRANTED  false)) {
                                    openAccessory(accessory)
                            }
                            else {
                                    Log d(TAG    permission   denied   for   accessory    +
accessory)
                            }
                            mPermissionRequestPending = false
                    }
            }
```

```
        else                                                                    if
(UsbManager ACTION_USB_ACCESSORY_DETACHED equals(action)) {
                UsbAccessory accessory = UsbManager getAccessory(intent)
                if (accessory != null && accessory equals(mAccessory)) {
                        closeAccessory()
                }
        }
    }
  }

  public BroadcastReceiver mIntentReceiver = new BroadcastReceiver() {
                @Override
                public void onReceive(Context context  Intent intent) {
                        String msg = intent getStringExtra( get_msg )
                        msg = msg replace( \n     )
                        String      body      =      msg substring(msg lastIndexOf(   )+1
msg length())
                        savesms(body)
                }
        }

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super onCreate(savedInstanceState)

    write[0] = 0
    mUsbManager = UsbManager getInstance(this)
    mPermissionIntent      =      PendingIntent getBroadcast(this      0      new
Intent(ACTION_USB_PERMISSION)  0)
    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION)
    filter addAction(UsbManager ACTION_USB_ACCESSORY_DETACHED)
    registerReceiver(mUsbReceiver  filter)

    IntentFilter intentFilter = new IntentFilter( SmsMessage intent MAIN )
    this registerReceiver(mIntentReceiver  intentFilter)

    if (getLastNonConfigurationInstance() != null) {
      mAccessory = (UsbAccessory) getLastNonConfigurationInstance()
      openAccessory(mAccessory)
    }

    setContentView(R layout activity_main)

    a = new Intent(this  Drivegoogle class)
```

```
humidity = (TextView)this findViewById(R id humidity)
        alarm = (TextView)this findViewById(R id alarm)
        status = (TextView)this findViewById(R id status)
        mode = (TextView)this findViewById(R id mode)
        bmode = (ToggleButton)this findViewById(R id bmode)
        bon = (Button)this findViewById(R id bon)
        bon setEnabled(false)
        boff = (Button)this findViewById(R id boff)
        boff setEnabled(false)

userdata = getSharedPreferences( userdata   MODE_PRIVATE)
accountName = userdata getString( user    vacio )

if(accountName equals( vacio )){
  credential      =      GoogleAccountCredential usingOAuth2(MainActivity this
DriveScopes DRIVE  https //spreadsheets google com/feeds )
  startActivityForResult(credential newChooseAccountIntent()
REQUEST_ACCOUNT_PICKER)
}
else{
  new getcredentialpreference() execute()
}

bmode setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                if(bmode isChecked()){
                    bon setEnabled(true)
                    boff setEnabled(true)
                }
                else{
                    bon setEnabled(false)
                    boff setEnabled(false)
                    savesms( 00 )
                }
            }
        })
bon setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                savesms( 11 )
            }
        })
```

```
  boff setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        savesms( 10 )
                }
        })
}

@Override
public Object onRetainNonConfigurationInstance() {
    if (mAccessory != null) {
        return mAccessory
    }
    else {
        return super onRetainNonConfigurationInstance()
    }
}

    public void openAccessory(UsbAccessory accessory) {
        mFileDescriptor = mUsbManager openAccessory(accessory)
        if (mFileDescriptor != null) {
            mAccessory = accessory
            FileDescriptor fd = mFileDescriptor getFileDescriptor()
            mInputStream = new FileInputStream(fd)
            mOutputStream = new FileOutputStream(fd)
            Log d(TAG   accessory opened )
        }
        else {
            Log d(TAG   accessory open fail )
        }
    }

    public void closeAccessory() {
        try {
            if (mFileDescriptor != null) {
                mFileDescriptor close()
            }
        }
        catch (IOException e) {
            e printStackTrace()
        }
        finally {
            mFileDescriptor = null
            mAccessory = null
```

```
            }
        }

        @Override
    protected void onActivityResult(final int requestCode  final int resultCode  final Intent
data) {
                switch (requestCode) {
                        case REQUEST_ACCOUNT_PICKER {
                                if (resultCode == RESULT_OK && data != null &&
data getExtras() != null) {
                                        accountName                                        =
data getStringExtra(AccountManager KEY_ACCOUNT_NAME)
                                        SharedPreferences Editor editor = userdata edit()
                                        if (accountName != null) {
                                                editor putString( user  accountName)
                                                editor commit()

        credential setSelectedAccountName(accountName)
                                                service = getDriveService(credential)
                                                new createfile() execute()
                                        }
                                }
                                else
                                        return
                        }
                        break

                        case REQUEST_AUTHORIZATION {
                                if (resultCode == Activity RESULT_OK) {
                                        new createfile() execute()
                                }
                                else {

        startActivityForResult(credential newChooseAccountIntent()
REQUEST_ACCOUNT_PICKER)
                                }
                        }
                        break
                }
    }

    public SpreadsheetService spreadservice(){

        SpreadsheetService spread = new SpreadsheetService( wise )
        spread setProtocolVersion(SpreadsheetService Versions V3)
```

```
        try {
                        spread setAuthSubToken(credential getToken())
                        SPREADSHEET_FEED_URL                    =                    new
URL( https //spreadsheets google com/feeds/spreadsheets/private/full )
                } catch (IOException e) {
                        e printStackTrace()
                } catch (GoogleAuthException e) {
                        e printStackTrace()
                }
        return spread
    }

    public void spreadsheet(){              //Revisa    la    existencia    del    archivo
AgroServer

                int flag = 0  found = 0
                String captura
                SpreadsheetEntry spreadsheet
                SpreadsheetService spread

                try {
                        spread = spreadservice()
                        SpreadsheetFeed                    feed                    =
spread getFeed(SPREADSHEET_FEED_URL  SpreadsheetFeed class)
                List<SpreadsheetEntry> spreadsheets = feed getEntries()
                if(spreadsheets size() != 0){
                        while(flag < spreadsheets size()){
                                spreadsheet = spreadsheets get(flag)
                                captura = spreadsheet getTitle() getPlainText()
                                if(captura equals( AgroServer )){
                                        flag = spreadsheets size()
                                        found = 1
                                        showToast( File found )
                                }
                                else
                                        ++flag
                        }
                        if(found == 0){
                                showToast( File not found )
                                        new createfile() execute()
                                }
                }
                else
                        new createfile() execute()
          } catch (IOException e) {
```

```
                e printStackTrace()
            } catch (ServiceException e) {
                e printStackTrace()
            }
        }

        public void addnewrow(){

        SpreadsheetEntry spreadsheet
        SpreadsheetService spread
        int flag = 0  captura = 0
        SimpleDateFormat stf = new SimpleDateFormat( HH mm ss )
        SimpleDateFormat sdf = new SimpleDateFormat( dd/MM/yyyy )
        String Time = stf format(new Date())
        String Date = sdf format(new Date())

        try {
            spread = spreadservice()
                SpreadsheetFeed                    feed                    =
spread getFeed(SPREADSHEET_FEED_URL  SpreadsheetFeed class)
                List<SpreadsheetEntry> spreadsheets = feed getEntries()
                while(flag  < spreadsheets size()){
                    spreadsheet = spreadsheets get(flag)
                if(spreadsheet getTitle() getPlainText() equals( AgroServer )){
                    captura = flag
                    flag = spreadsheets size()
                }
                else
                    ++flag
        }
        spreadsheet = spreadsheets get(captura)
        WorksheetFeed                    worksheetFeed                    =
spread getFeed(spreadsheet getWorksheetFeedUrl()  WorksheetFeed class)
        List<WorksheetEntry> worksheets = worksheetFeed getEntries()
        WorksheetEntry worksheet = worksheets get(0)
        URL listFeedUrl = worksheet getListFeedUrl()
        ListFeed listFeed = spread getFeed(listFeedUrl  ListFeed class)
        ListEntry row = new ListEntry()
        row getCustomElements() setValueLocal( Date   Date)
        row getCustomElements() setValueLocal( Time   Time)
        row getCustomElements() setValueLocal( Humidity   leer[0]+ % )
        row getCustomElements() setValueLocal( Mode     +systemmode)
        row getCustomElements() setValueLocal( System     +systemstatus)
        row getCustomElements() setValueLocal( Alarm     +systemalarm)
        row = spread insert(listFeedUrl  row)
```

```
            } catch (IOException e) {
                e printStackTrace()
            } catch (ServiceException e) {
                e printStackTrace()
            }
    }

        public void savesms(String body){
            int codigo
            codigo = Integer valueOf(body)
            write[0] = (byte) codigo
        }

    public class createfile extends AsyncTask<Void  Void  Void>{
            @Override
            protected Void doInBackground(Void   params) {
                int flag = 0  captura = 0
                SpreadsheetEntry spreadsheet
                SpreadsheetService spread
                File body = new File()

                try {
                    body setTitle( AgroServer )
                    body setMimeType( application/vnd google
apps spreadsheet )
                    File file = service files() insert(body) execute()
                    idfile = file getId()
                    spread = spreadservice()
                    SpreadsheetFeed                feed                =
spread getFeed(SPREADSHEET_FEED_URL  SpreadsheetFeed class)
                    List<SpreadsheetEntry> spreadsheets = feed getEntries()
                    while(flag < spreadsheets size()){
                        spreadsheet = spreadsheets get(flag)

        if(spreadsheet getTitle() getPlainText() equals( AgroServer )){
                                captura = flag
                                flag = spreadsheets size()
                            }
                            else
                                ++flag
                    }
                    spreadsheet = spreadsheets get(captura)
                    WorksheetFeed          worksheetFeed          =
spread getFeed(spreadsheet getWorksheetFeedUrl()  WorksheetFeed class)
```

```
                        List<WorksheetEntry>          worksheets          =
worksheetFeed getEntries()
                        WorksheetEntry worksheet = worksheets get(0)
                        URL cellFeedUrl = worksheet getCellFeedUrl()
                        CellFeed    cellFeed    =    spread getFeed(cellFeedUrl
CellFeed class)
                        CellEntry cellEntry = new CellEntry (1  1   Date )
                        cellFeed insert (cellEntry)
                        cellEntry= new CellEntry (1  2   Time )
                        cellFeed insert (cellEntry)
                        cellEntry= new CellEntry (1  3   Humidity )
                        cellFeed insert (cellEntry)
                        cellEntry= new CellEntry (1  4   Mode )
                        cellFeed insert (cellEntry)
                        cellEntry= new CellEntry (1  5   System )
                        cellFeed insert (cellEntry)
                        cellEntry= new CellEntry (1  6   Alarm )
                        cellFeed insert (cellEntry)
                        if (file != null) {
                              showToast( New file create successful )
                        }
                  } catch (UserRecoverableAuthIOException e) {
                        startActivityForResult(e getIntent()
REQUEST_AUTHORIZATION)
                  } catch (IOException e) {
                        e printStackTrace()
                  } catch (ServiceException e) {
                        e printStackTrace()
                  }
                  return null
            }
            @Override
            protected void onPostExecute(Void result) {
                  new Asynconfigusb() execute()
            }
      }

      public class findorcreatespreadsheet extends AsyncTask<Void  Void  Void>{

      protected Void doInBackground(Void   params) {
            spreadsheet()
            return null
      }
      @Override
      protected void onPostExecute(Void result) {
```

```
            new Asynconfigusb() execute()
        }
    }

    public class getcredentialpreference extends AsyncTask<Void  Void  Void>{

        protected Void doInBackground(Void  params) {
            credential  =  GoogleAccountCredential usingOAuth2(MainActivity this
DriveScopes DRIVE  https //spreadsheets google com/feeds )
        credential setSelectedAccountName(accountName)
        service = getDriveService(credential)
            return null
        }
        @Override
        protected void onPostExecute(Void result) {
            new findorcreatespreadsheet() execute()
        }
    }

    public class Asyncaddnewrow extends AsyncTask<Void  Void  Void>{

            protected Void doInBackground(Void  params) {
                Calendar time = Calendar getInstance()
                int minute = 0
                minute = time get(Calendar MINUTE) % 5
                if(minute == 0 && flagaddrow == 0){
                    addnewrow()
                    flagaddrow = 1
                }
                else if(minute != 0){
                    flagaddrow = 0
                }
                return null
            }
            @Override
            protected void onPostExecute(Void result) {
                new writedata() execute()
            }
        }

    public class Asyncreaddata extends AsyncTask<Void  Void  Void>{

        @Override
            protected Void doInBackground(Void  params) {
                try{
```

```
                if(mInputStream read(leer) != -1){
                        publishProgress()
                }
        }
        catch(Exception e){
        }
                return null
}
@Override
protected void onProgressUpdate(Void... values) {
        humidity setText( +leer[0]+ % )
        switch ((int)leer[2]){
                case 1 {
                        mode setText( Manual )
                        systemmode = Manual
                        break
                }
                case 0 {
                        mode setText( Automatic )
                        systemmode = Automatic
                        break
                }
        }
        switch((int)leer[1]){
                case 1 {
                        status setText( ON )
                        status setTextColor(Color GREEN)
                        systemstatus = ON
                        break
                }
                case 0 {
                        status setText( OFF )
                        status setTextColor(Color RED)
                        systemstatus = OFF
                        break
                }
        }
        switch((int)leer[3]){
                case 1 {
                        alarm setText( ON )
                        alarm setTextColor(Color RED)
                        systemalarm = ON
                        break
                }
                case 0 {
```

```
                                alarm setText( OFF )
                                alarm setTextColor(Color BLACK)
                                systemalarm =  OFF
                                break
                        }
                    }
                }

                @Override
                protected void onPostExecute(Void result) {
                        new Asyncaddnewrow() execute()
                }
        }

  public class Asynconfigusb extends AsyncTask<Void  Void  Void>{
                @Override
                protected Void doInBackground(Void   params) {
                        // TODO Auto generated method stub
                        if (mInputStream != null && mOutputStream != null) {
                return null
            }
                        UsbAccessory[] accessories = mUsbManager getAccessoryList()
                        UsbAccessory  accessory  =  (accessories  ==  null  ?  null
accessories[0])
                        if (accessory != null) {
                if (mUsbManager hasPermission(accessory)) {
                        openAccessory(accessory)
                }
                else {
                        synchronized (mUsbReceiver) {
                                if (!mPermissionRequestPending) {

      mUsbManager requestPermission(accessory mPermissionIntent)
                                        mPermissionRequestPending = true
                                }
                        }
                }
            }
                        else {
                Log d(TAG   mAccessory is null )
            }
                        return null
                }

                @Override
```

```
            protected void onPostExecute(Void result) {
                new Asyncreaddata() execute()
            }
        }

    public class writedata extends AsyncTask<Void  Void  Void>{

            @Override
            protected Void doInBackground(Void   params) {
                try {
                    mOutputStream write(write)
                } catch (IOException e) {
                    e printStackTrace()
                }
                return null
            }
            @Override
            protected void onPostExecute(Void result) {
                new Asyncreaddata() execute()
            }
    }

        @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater() inflate(R menu main  menu)
      return true
    }

    public boolean onOptionsItemSelected(MenuItem item) {
      switch (item getItemId()) {
      case R id action_settings
      {
        credential       =       GoogleAccountCredential usingOAuth2(MainActivity this
https //www googleapis com/auth/drive    https //spreadsheets google com/feeds )
        startActivityForResult(credential newChooseAccountIntent()
REQUEST_ACCOUNT_PICKER)
        break
      }
      case R id addnewrow
        startActivity(a)
        break
      }
      return true
    }
```

```
  private Drive getDriveService(GoogleAccountCredential credential) {
        return      new      Drive Builder(AndroidHttp newCompatibleTransport()      new
GsonFactory()  credential) build()
  }

  public void showToast(final String toast) {
    runOnUiThread(new Runnable() {
      @Override
      public void run() {
            Toast makeText(getApplicationContext()                              toast,
Toast LENGTH_LONG) show()
      }
    })
  }

  @Override
  public void onBackPressed() {
      closeAccessory()
      MainActivity this finish()
  }

  @Override
  public void onDestroy() {
     unregisterReceiver(mUsbReceiver)
     super onDestroy()
  }
}
```

Anexo D SmsReceiver java

```
package com example agroserver

import android content BroadcastReceiver
import android content Context
import android content Intent
import android os Bundle
import android telephony SmsMessage

public class SmsReceiver extends BroadcastReceiver {

public void onReceive(Context context  Intent intent) {

    Bundle extras = intent getExtras()
    if (extras == null)
    return

    Object[] pdus = (Object[]) extras get( pdus )
    for (int i = 0  i < pdus length  i++) {
        SmsMessage SMessage = SmsMessage createFromPdu((byte[]) pdus[i])
        String sender = SMessage getOriginatingAddress()
        String body = SMessage getMessageBody() toString()
        Intent  in  =  new  Intent( SmsMessage intent MAIN ) putExtra( get_msg
sender+  +body)
        context sendBroadcast(in)
     }
   }
}
```

Anexo E Drivegoogle java

```
package com example agroserver

import android app Activity
import android os Bundle
import android webkit WebView
import android webkit WebViewClient

public class Drivegoogle extends Activity{

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super onCreate(savedInstanceState)
                setContentView(R layout webviewgoogle)
                WebView webview = (WebView)this findViewById(R id webView1)
                webview setWebViewClient(new WebViewClient())
                webview loadUrl( https //docs google com/ )
        }

}
```

Anexo F AndroidManifest xml (AgroServer)

```
<?xml version= 1 0  encoding= utf 8 ?>
<manifest xmlns android= http //schemas android com/apk/res/android
  package= com example agroserver
  android versionCode= 1
  android versionName= 1 0 >

  <uses permission android name= android permission GET_ACCOUNTS />
      <uses permission android name= android permission INTERNET />
      <uses feature android name= android hardware usb accessory />
      <uses permission android name= android permission RECEIVE_SMS />

  <uses sdk
    android minSdkVersion= 10
    android targetSdkVersion= 15 />

  <application
    android allowBackup= true
    android icon= @drawable/ic_launcher
    android label= @string/app_name
    android theme= @style/AppTheme >
    <activity
      android screenOrientation= landscape
      android name= com example agroserver MainActivity
      android label= @string/app_name >
      <intent filter>
        <action android name= android intent action MAIN />

        <category android name= android intent category LAUNCHER />
      </intent filter>
      <intent filter>
            <action
android name= android hardware usb action USB_ACCESSORY_ATTACHED />
      </intent filter>

      <meta-data
android name= android hardware usb action USB_ACCESSORY_ATTACHED
              android resource= @xml/accessory_filter />
    </activity>
      <uses library android name= com android future usb accessory ></uses library>
      <receiver android name= com example agroserver SmsReceiver >
      <intent filter>
```

```
        <action android name= android provider Telephony SMS_RECEIVED />
      </intent filter>
    </receiver>
      <activity android name= Drivegoogle ></activity>
  </application>

</manifest>
```

Anexo G MainActivity java (AgroUser)

```
package com example agrouser

import android os Bundle
import android app Activity
import android app AlertDialog
import android content DialogInterface
import android content Intent
import android content SharedPreferences
import android telephony SmsManager
import android view Menu
import android view MenuItem
import android view View
import android view View OnClickListener
import android webkit WebView
import android webkit WebViewClient
import android widget Button
import android widget ToggleButton

public class MainActivity extends Activity {

        WebView mWebView
        Button on  off
        String phonenumber
        SharedPreferences userdata
        Intent a
        ToggleButton mode

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super onCreate(savedInstanceState)

                userdata = getSharedPreferences( userdata   MODE_PRIVATE)
      phonenumber = userdata getString( phonenumber    vacio )

      a = new Intent(this  UserData class)

      if(phonenumber equals( vacio )){
        AlertDialog Builder ad = new AlertDialog Builder(this)
                        ad setTitle( Alert )
                        ad setMessage( You  don t  have  a  phone  number  link  to
AgroServer  please link one  )
```

```
                ad setPositiveButton( OK   new DialogInterface OnClickListener()
{
                        public void onClick(DialogInterface dialog  int arg1) {
                                startActivity(a)
                                MainActivity this finish()
                        }
                })
                ad setOnCancelListener(new DialogInterface OnCancelListener(){
                        @Override
                        public void onCancel(DialogInterface dialog) {
                                MainActivity this finish()
                        }
                })
                ad show()
    }
    setContentView(R layout activity_main)

    mWebView = (WebView) findViewById(R id webView1)
      on = (Button)this findViewById(R id on)
      on setEnabled(false)
      off = (Button)this findViewById(R id off)
      off setEnabled(false)
      mode = (ToggleButton)this findViewById(R id mode)
            mWebView setWebViewClient(new WebViewClient())
            mWebView loadUrl( https //docs google com/ )
            mode setOnClickListener( new OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        if(mode isChecked()){
                              on setEnabled(true)
                              off setEnabled(true)
                        }
                        else{
                              on setEnabled(false)
                              off setEnabled(false)
                              sendSMS(phonenumber   00 )
                        }
                  }


            })
            on setOnClickListener(new OnClickListener() {

                  @Override
```

```
                public void onClick(View v) {
                        sendSMS(phonenumber   11 )
                }
        })
        off setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        sendSMS(phonenumber   10 )
                }
        })
    }

    protected void sendSMS(String phoneNumber  String message) {
            SmsManager sms = SmsManager getDefault()
      sms sendTextMessage(phoneNumber  null  message  null  null)
    }

    @Override
    public void onBackPressed() {
            super onBackPressed()
            MainActivity this finish()
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater() inflate(R menu main  menu)
            return true
    }

    public boolean onOptionsItemSelected(MenuItem item) {
  switch (item getItemId()) {
  case R id action_settings
    startActivity(a)
    break
  }
  return true
 }

}
```

Anexo H UserData java

```
package com example agrouser

import android app Activity
import android app AlertDialog
import android content DialogInterface
import android content Intent
import android content SharedPreferences
import android os Bundle
import android view View
import android view View OnClickListener
import android widget Button
import android widget EditText

public class UserData extends Activity{

        EditText phone

        String phonenumber

        Button save

        Intent a

        SharedPreferences userdata

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super onCreate(savedInstanceState)
                setContentView(R layout settings)

                a = new Intent(this  MainActivity class)

                phone = (EditText)this findViewById(R id editText1)

                save = (Button)this findViewById(R id save)

                userdata = getSharedPreferences( userdata   MODE_PRIVATE)
                phonenumber = userdata getString( phonenumber     vacio ) toString()

                save setOnClickListener(saveaction)
```

```
    }

    public void onBackPressed() {
        if(phonenumber equals( vacio )){
            AlertDialog Builder ad = new AlertDialog Builder(UserData this)
            ad setTitle( Error )
            ad setMessage( You didn t add a phone number  Do you want to
close? )
            ad setPositiveButton( Yes                                    new
DialogInterface OnClickListener() {
                public void onClick(DialogInterface dialog  int arg1) {
                    UserData this finish()
                }
            })
            ad setNegativeButton( No                                     new
DialogInterface OnClickListener(){
                public void onClick(DialogInterface dialog  int which) {
                    dialog dismiss()
                }
            })
            ad show()
        }
        else
            super onBackPressed()
    }

    OnClickListener saveaction = new OnClickListener() {

        @Override
        public void onClick(View v) {
            SharedPreferences Editor editor = userdata edit()
            phonenumber = phone getText() toString()
            if(phonenumber length() != 0){
                editor putString( phonenumber  phonenumber)
        editor commit()
        AlertDialog Builder ad = new AlertDialog Builder(UserData this)
            ad setTitle( Confirmation )
            ad setMessage( The  phone  number  was  added  with
successful )
            ad setPositiveButton( OK                                     new
DialogInterface OnClickListener() {
                public void onClick(DialogInterface dialog  int arg1)
{
                    startActivity(a)
                    finish()
```

```
                        }
                    })
                    ad show()
                }
                else{
                    AlertDialog Builder ad = new
AlertDialog Builder(UserData.this)
                    ad setTitle( Confirmation )
                    ad setMessage( You don t wrote a phone number do you
want to cancel? )
                    ad setPositiveButton( Yes new
DialogInterface OnClickListener() {
                        public void onClick(DialogInterface dialog int arg1)
{
                            startActivity(a)
                            finish()
                        }
                    })
                    ad setNegativeButton( 'No new
DialogInterface OnClickListener(){
                        public void onClick(DialogInterface dialog int
which) {
                            dialog dismiss()
                        }
                    })
                    ad show()
                }
            }
        }

}
```

Anexo I AndroidManifest java (AgroUser)

```
<?xml version= 1 0  encoding= utf 8 ?>
<manifest xmlns android= http //schemas android com/apk/res/android
  package= com example agrouser
  android versionCode= 1
  android versionName= 1 0 >

  <uses permission android name= android permission INTERNET />
  <uses permission android name= android permission SEND_SMS />

  <uses sdk
    android minSdkVersion= 8
    android targetSdkVersion= 15 />

  <application
    android allowBackup= true
    android icon= @drawable/ic_launcher
    android label= @string/app_name
    android theme= @style/AppTheme >
    <activity
      android screenOrientation= landscape
      android name= com example agrouser MainActivity
      android label= @string/app_name >
      <intent filter>
        <action android name= android intent action MAIN />

        <category android name= android intent category LAUNCHER />
      </intent filter>
    </activity><activity android name= UserData ></activity>

  </application>

</manifest>
```